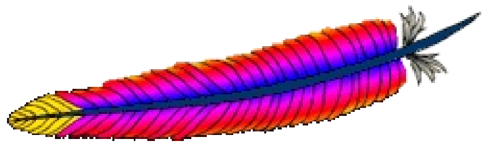
Apache 2.2 中文官方文档

wizardforcel


Published with GitBook

目錄

Apache 中文文档 2.2



译者：金步国

来源：Apache 2.2 中文手册

Apache HTTP Server Version 2.2 文档 [最后更新：2006年3月21日]

版本说明

- Apache 2.1/2.2 版本的新特性
- Apache 2.0 版本的新特性
- 从 2.0 升级到 2.2
- Apache许可证

参考手册

- 编译与安装
- 启动
- 停止与重新启动
- 运行时配置指令
- 指令速查
- 模块索引
- 多路处理模块(MPM)
- 过滤器
- 处理器
- 服务器与工具
- 词汇表

用户指南

- 地址和端口绑定
- 配置文件
- 配置段
- 内容缓冲
- 内容协商
- 动态共享对象(DSO)
- 环境变量
- 日志文件
- 从URL到文件系统的映射
- 性能调整
- 安全方面的提示

- 服务器全局配置
- SSL/TLS 加密
- CGI脚本的Suexec执行
- URL重写指南
- 虚拟主机

如何.../指南

- 认证、授权、访问控制
- CGI动态网页
- .htaccess 文件
- 服务器端包含(SSI)
- 用户网站目录的管理(public_html)

对特定平台的说明

- Microsoft Windows
- Novell NetWare
- EBCDIC Port

其他

- 常见问题解答
- 站点导航
- 开发者文档
- 其他文档

版本说明

从1.3升级到2.0

为了帮助大伙儿升级，我们为现在的Apache用户提供了一份重要信息的文档说明。这些只是一些简要说明，你可以从新特性文档或 `src/CHANGES` 文件中得到更多信息。

编译时配置的改变

- Apache现在使用autoconf和libtool系统来进行安装进程的配置。这个系统用起来很像Apache1.3的APACI系统，但并不相同。
- 在普通的选择编译模块的基础上，Apache2.0把请求进程的主要部分移到了多路处理模块(MPM)里。

运行时配置的改变

- Apache1.3服务器核心中的很多指令现在都放到了MPM里面。如果你希望服务器的行为能够尽量的类似于Apache1.3，你应当选择 `prefork` MPM。其他的MPM将拥有不同的指令来控制进程创建和请求过程。
- proxy module已经被修补以兼容HTTP/1.1。其中重要的改变之一是：代理的访问控制现在是放在 `<Proxy>` 段而不是 `<Directory proxy:>` 段里面了。
- 许多模块中 `PATH_INFO` (在真实文件名后附加路径信息)的处理有了变化。以前作为处理器而现在作为过滤器出现的模块现在可能不再接受包含 `PATH_INFO` 的请求。诸如INCLUDES或PHP过滤器将在处理核心的最顶层得到实现，从而拒绝包含 `PATH_INFO` 的请求。你可以用 `AcceptPathInfo` 指令来迫使处理核心接受包含 `PATH_INFO` 的请求，从而恢复服务器端包含中使用 `PATH_INFO` 的能力。
- `CacheNegotiatedDocs` 指令现在使用 On 或 Off 参数了。原有的 `CacheNegotiatedDocs` 应该代之以 `CacheNegotiatedDocs on`
- `ErrorDocument` 指令不再用引号开始的参数来指定文本内容了。取而代之的是用双引号把文本内容括起来。比如原有的配置：

  ```
  ErrorDocument 403 "Some Message
  ```

  应该代之以：

  ```
  ErrorDocument 403 "Some Message"
  ```

  只要第二个参数不是有效的URL或路径名，它就会被当作是一个文本信息。

- `AccessConfig` 和 `ResourceConfig` 指令不复存在了。现有的这些指令可以用 `Include` 指令

代替以实现相同的功能。如果你使用的是这些指令的默认值而没有把它们放到配置文件里的话，你可能需要把" `Include conf/access.conf` "和" `Include conf/srm.conf` "加到你的 `httpd.conf` 里。为了确保Apache用象以前一样的顺序读取这些配置文件，应该把 `Include` 指令放到 `httpd.conf` 的结束部分，并将包含 `srm.conf` 的语句放在包含 `access.conf` 的语句的前面。

- `BindAddress` 和 `Port` 指令不再存在了。相同的功能由更加灵活的 `Listen` 指令提供。
- Apache1.3中 `Port` 指令的另一功能是设定自引用的URL的端口。Apache2.0中对等的是新的 `ServerName` 语法：它已经被修改成在一条指令里同时为自引用的URL指定服务器名和端口号。
- `ServerType` 指令不复存在了。用于伺服请求的方法现在取决于MPM的选择。目前还没有设计出用于被inetd(端口监视程序)载入的MPM。
- `mod_log_agent` 和 `mod_log_referer` 被去掉了。取代以使用 `CustomLog` 指令的 `mod_log_config` 模块。
- `AddModule` 和 `ClearModuleList` 指令不复存在了。这些指令原用于确定模块以正确的顺序被激活。而新的Apache2.0 API允许模块明确的指定它们的顺序，从而这些指令就不再有存在的必要了。
- `FancyIndexing` 指令被去掉了，取而代之的是 `IndexOptions` 指令的 `FancyIndexing` 选项。
- 由模块 `mod_negotiation` 提供的MultiViews内容协商机制在其默认文件匹配方面变得更加严格了，只匹配允许协商的文件。可以用 `MultiviewsMatch` 指令恢复到原来的匹配模式。
- (*2.0.51*以后) `ErrorHeader` 指令的功能合并到 `Header` 指令中去了。因为原来的是一个谬误。应当使用：

  ```
  Header always set foo bar
  ```

  代替原来的使用方式。

杂项的改变

- Apache1.3中的实验模块 `mod_auth_digest` 现在是基本模块了。
- Apache1.3中的实验模块 `mod_mmap_static` 现在被 `mod_file_cache` 代替了。
- 发行包经过了重新组织，从而不再包含一个独立的src目录。取而代之的是将源代码有逻辑的组织在发行包的主目录下面。编译后的服务器的安装将从各自的目录下进行。

第三方模块

Apache2.0中的服务器API有了巨大的变化。现有的为Apache1.3设计的模块未经修改将不能运行在Apache2.0上。详情请参见开发者文档。

从2.0升级到2.2

为了帮助大伙儿升级，我们为现在的Apache用户提供了一份重要信息的文档说明。这些只是一些简要说明，你可以从新特性文档或 `src/CHANGES` 文件中得到更多信息。

这篇文档仅仅描述了从版本 2.0 到 2.2 的变化，如果你是从1.3版进行升级的，请查考从1.3升级到2.0文档。

编译时配置的改变

编译过程与2.0版本非常相似，你曾经使用过的 `configure` 命令行(在安装目录下的 `build/config.nice` 文件中)在某些情况下仍然可以使用。主要是模块的名称变化了，特别是认证和授权模块。具体如下：

- mod_imap 被更名为 `mod_imagemap`
- mod_auth 被拆分为 `mod_auth_basic` 、 `mod_authn_file` 、 `mod_authz_user` 、 `mod_authz_groupfile`
- mod_access 被更名为 `mod_authz_host`
- mod_auth_ldap 被更名为 `mod_authnz_ldap`
- 需要使用 APR 1.0 API
- 将内置的PCRE升级到5.0版本

运行时配置的改变

除了下面讨论的某些特定配置需要进行很小的调整以外，现存的2.0版本的配置文件和启动脚本基本上不需要修改就可以直接用在2.2版本中。另外，如果你使用 `LoadModule` 动态加载了标准模块。你还需要根据上面提到的模块名变更情况适当修改原有配置文件。

如果你选择使用新的2.2版默认配置文件，你将会发现它已经删除了绝大多数不属于基本配置的指令，比以前大大简化了。在安装目录下的 `conf/extra/` 子目录中，有一组包含许多高级特性的示例配置。默认的配置文件被安装在 `conf/original` 子目录中。

一些需要注意的运行时配置更改：

- `apachectl` 选项 `startssl` 被取消了。要启用SSL支持，你必须编辑 `httpd.conf` 文件，在其中包含与 `mod_ssl` 相关的指令，然后使用 `apachectl start` 命令启动服务器。我们提供了一个示范如何启用 `mod_ssl` 的配置文件： `conf/extra/httpd-ssl.conf` 。
- 现在， `UseCanonicalName` 的默认值是 `Off` 。如果原来的配置文件中没有使用这个指令，你可以在其中添加 `UseCanonicalName On` 以保持和原来的行为相同。
- 除非明确使用 `UserDir` 指令在配置文件中指定了一个目录，否则， `mod_userdir` 模块将不

会被激活。若想恢复到与原来默认行为相同，你必须在配置文件中添加一行"`UserDir public_html`"。

杂项变化

- `mod_cache` 原来在2.0中是试验模块，现在变成标准模块了。
- `mod_disk_cache` 原来在2.0中是试验模块，现在变成标准模块了。
- `mod_mem_cache` 原来在2.0中是试验模块，现在变成标准模块了。
- `mod_charset_lite` 原来在2.0中是试验模块，现在变成标准模块了。
- `mod_dumpio` 原来在2.0中是试验模块，现在变成标准模块了。

第三方模块

大多数2.0版本的第三方模块可以不加修改的运行于2.2版本中。但是这些模块必须要重新进行编译后才能使用。

Apache 2.2 新特性概述

这篇文档简述了Apache HTTP Server 2.0和2.2 版本之间的主要差异。对于1.3版本以后的新特性，请参考Apache 2.0 新特性文档。

核心增强

认证/授权(Authn/Authz)

原本捆绑在一起的认证(authentication)与授权(authorization)模块现在被分开了。新的 `mod_authn_alias` 模块可以极大的简化某些身份认证的配置。请参见模块名的变更和针对模块开发者的变更以了解更多有关这些变更对于模块使用者和模块开发者的影响。

缓冲

`mod_cache` 、 `mod_disk_cache` 、 `mod_mem_cache` 经历了诸多修改以后现在已经具备了合格的产品质量了。新增加的 `htcacheclean` 工具可以用来清理 `mod_disk_cache` 模块使用的缓冲存储区。

配置

默认的配置布局已经被简化并模块化了。启用常用特性的配置片段现在已经和Apache捆绑在一起，可以被轻易的添加到主配置文件中。

优雅停止(Graceful stop)

`prefork` 、 `worker` 、 `event` 多路处理模块(MPM)现在允许 `httpd` 通过 `graceful-stop` 信号被优雅的停止。可以用新增的 `GracefulShutdownTimeout` 指令指定一个超时时间，超过指定的时间以后 `httpd` 将会强行中止，而无论请求所处的服务状态如何。

代理

新增的 `mod_proxy_balancer` 模块为 `mod_proxy` 提供了负载均衡服务。新增的 `mod_proxy_ajp` 模块为Apache Tomcat使用的 `Apache JServ Protocol version 1.3` 提供了支持。

正则表达式库更新

5.0版的Perl兼容正则表达式库(PCRE)已经被包含进来了。 `httpd` 现在可以通过 `--with-pcre` 编译选项使用系统中已经安装好的PCRE。

智能过滤器

`mod_filter` 可以进行输出过滤器链的动态配置。它允许过滤器按照请求头或应答头或环境变量有条件的插入，这样就避免了许多在Apache2.0体系结构中存在的过滤器之间的依赖性和顺序问题。

大文件支持

httpd现在已经被构建为在现代的32位Unix系统上支持大于2GB的文件。而且也可以处理大于2G的请求体(request body)。

Event MPM

`event` 多路处理模块(MPM)使用一个单独隔开的线程处理持久连接(Keep Alive)。传统上，持久连接要求httpd专门拿出一个工作者(worker)(也就是一个进程/线程)来处理它。这个专用的工作者在持久连接超时前不能被重新使用。

SQL数据库支持

`mod_dbd` 和apr_dbd框架(framework)一起为需要使用数据库的模块提供直接的支持。在线程化的MPM中还能支持连接缓冲池。

Windows用户请注意，这个特性尚未包含在标准的windows版Apache中。如果你尝试在Windows平台上使用这个特性，请告诉我们你的进展情况。

模块增强

认证/授权(Authn/Authz)

aaa目录下的模块已经被重新命名并提供了对摘要认证(digest authentication)的更好支持。例如，mod_auth现在已经被分割成 `mod_auth_basic` 和 `mod_authn_file` 两个模块；mod_auth_dbm现在更名为 `mod_authn_dbm` ；mod_access现在更名为 `mod_authz_host` ；还新增了一个 `mod_authn_alias` 模块用于简化某些认证配置。

`mod_authnz_ldap`

这个模块是2.0版 `mod_auth_ldap` 模块到2.2版的 `Authn/Authz` 框架的一个移植。新的特性包括使用LDAP属性值和 `Require` 指令中复杂的搜索过滤器。

`mod_info`

添加了一个新的 `?config` 参数，可以用来显示被Apache分析过的配置指令，包括它们的文件名和行号。该模块还显示所有请求钩子(request hook)的顺序和额外的编译信息，有些类似于 `httpd -V`

。

`mod_ssl`

添加了RFC 2817支持，它允许连接从明文提升到TLS加密。

`mod_imagemap`

mod_imap已经被重命名为 `mod_imagemap` ，以避免用户产生混淆和疑惑。

程序增强

`httpd`

添加了一个新的命令行选项 `-M` 用来列出基于当前配置加载的所有模块。不同于 `-l` 选项的是，它还列出了通过 `mod_so` 加载的DSO(动态共享对象)。

`httxt2dbm`

一个用于从文本输入产生dbm文件的程序，目的是为了能够在 `RewriteMap` 中使用 `dbm` 映射表(map)类型。

针对模块开发者的变化

APR 1.0 API

Apache2.2 使用 APR 1.0 API 。所有反对使用的函数和符号已经从 `APR` 和 `APR-Util` 中清除掉了。欲知详情，请查看APR 网站。

认证/授权(Authn/Authz)

原来捆绑在一起的认证和授权模块已经被按照下列规则进行了重命名：

- mod_auth_* -> 实现HTTP认证机制的模块
- mod_authn_* -> 实现后端认证支持者的模块
- mod_authz_* -> 实现授权(或访问)的模块
- mod_authnz_* -> 同时实现认证和授权的模块

现在有一个新的认证后端提供者方案，可以简化新认证后端的创建。

连接错误日志

添加了一个新的 `ap_log_cerror` 函数用于记录客户端连接时发生的错误。并且在记录时包含客户端IP地址。

添加了一个测试配置的钩子(hook)

添加了一个新的 `test_config` 钩子，可以在用户向 `httpd` 传递 `-t` 选项时，执行包含特定代码的模块。

设置线程型MPM所使用的栈空间大小

新增的 `ThreadStackSize` 指令可以用来限制所有线程型MPM所使用的栈大小。一些默认栈空间较小的平台上的第三方模块需要使用它指定栈空间的大小。

输出过滤器协议处理

过去，每个过滤器都要确保自身能够产生正确的应答头。现在过滤器可以调用 `ap_register_output_filter_protocol` 或 `ap_filter_protocol` 来委托 `mod_filter` 进行协议管理。

添加了监视钩子(Monitor hook)

监视钩子可以让模块运行父进程中事先安排好的工作。

正则表达式 API 发生了变化

`pcreposix.h` 头文件现在被 `ap_regex.h` 头文件取代了。原来老的POSIX.2 `regex.h` 实现现在位于 `ap_` 名字空间下(由 `ap_regex.h` 提供)。比如原来的 `regcomp` , `regexec` 调用现在要修改成 `ap_regcomp` , `ap_regcomp` 调用。

DBD框架(SQL数据库API)

在1.x和2.0版本中，需要SQL支持的模块必须自己管理数据库。为了不要重新发明轮子，Apache 2.1 及以后的版本提供了 `ap_dbd` API 来管理数据库连接(包括对线程型和非线程型MPM进行优化)，同时 APR 1.2 及以后版本也提供了 `apr_dbd` API 与数据库打交道。

新模块应当使用了这些API来进行数据库操作。现存的应用程序应当进行透明的升级或使用推荐选项来使用这些API。

Apache 2.0 新特性概述

此文档描述了Apache 1.3和2.0版本之间的主要变化。

核心的增强

Unix线程

在支持POSIX线程的Unix系统上，现在Apache能在混合的多进程、多线程模式下运行，使很多(但非全部)配置的可伸缩性得到了改善。

新的编译系统

重写了编译系统，现在是基于 `autoconf` 和 `libtool` 的，使得Apache的配置系统与其他软件包更加相似。

多协议支持

Apache现在已经拥有了能够支持多协议的底层构造。 `mod_echo` 就是一个例子。

对非Unix平台更好的支持

Apache2.0在诸如BeOS、OS/2、Windows等非Unix平台上有了更好的速度和稳定性。随着平台特定的多路处理模块(MPM)和Apache可移植运行时(APR)的引入，Apache在这些平台上的指令由它们本地的API指令实现。避免了以往使用POSIX模拟层造成的bug和性能低下。

新的 Apache API

2.0中模块的API有了重大改变。很多1.3中模块排序和模块优先级的问题已经不复存在了。2.0自动处理了很多这样的问题，模块排序现在用per-hook的方法进行，从而拥有了更多的灵活性。另外，增加了新的调用以提高模块的性能，而无需修改Apache服务器核心。

IPv6 支持

在所有能够由Apache可移植运行时库(APR library)提供IPv6支持的系统上，Apache默认使用IPv6侦听套接字。另外， `Listen` 、 `NameVirtualHost` 、 `VirtualHost` 指令也支持IPv6的数字地址串(比如：" `Listen [2001:db8::1]:8080` ")。

过滤器

Apache的模块现在可以写成过滤器的形式，当内容流经它进入服务器或从服务器流出的时候进行处理。比如，可以用 `mod_include` 中的 `INCLUDES` 过滤器将CGI脚本的输出解析为服务器端包含指令。而 `mod_ext_filter` 允许外部程序充当过滤器的角色，就象用CGI程序做处理器一样。

多语种错误应答

返回给浏览器的错误信息现在已经用SSI文档实现了多语种化。管理员可以利用此功能进行定制以达到感观的一致。

简化了配置

很多易混淆的配置项已经进行了简化。经常产生混淆的 `Port` 和 `BindAddress` 配置项已经取消了；用于绑定IP地址的只有 `Listen` 指令； `ServerName` 指令中指定的服务器名和端口仅用于重定向和虚拟主机的识别。

本地 Windows NT Unicode 支持

Apache2.0在WindowsNT上的文件名全部使用utf-8编码。这个操作直接转换成底层的Unicode文件系统，由此为所有以WindowsNT(包括Windows2000/XP/2003)为基础的安装提供了多语言支持。这一支持目前尚未涵盖*Windows95/98/ME*系统，因为它们仍使用机器本地的代码页进行文件系统的操作。

正则表达式库更新

Apache2.0包含了Perl兼容的正则表达式库(PCRE)。所有正则表达式现在都使用了更强大的Perl 5 语法。

模块的增强

`mod_ssl`

Apache2.0中的新模块。此模块是OpenSSL提供的一个SSL/TLS加密协议接口。

`mod_dav`

Apache2.0中的新模块。此模块继承了HTTP分布式发布和版本控制规范，用于发布和维护web内容。

`mod_deflate`

Apache2.0中的新模块。此模块允许支持此功能的浏览器请求的页面内容在发送前进行压缩，以节省网络带宽。

`mod_auth_ldap`

Apache2.0.41中的新模块。此模块允许使用LDAP数据库存储HTTP基本认证所需的信息。随之而来的另一个模块 `mod_ldap` 则提供了连接池和结果的缓冲。

`mod_auth_digest`

利用共享内存实现了对跨进程会话缓冲的额外支持。

`mod_charset_lite`

Apache2.0中的新模块。这个试验模块允许在不同的字符集之间进行转换和重新编码。

`mod_file_cache`

Apache2.0中的新模块。这个模块包含了Apache1.3中 `mod_mmap_static` 模块的功能，另外进一步增加了缓冲能力。

`mod_headers`

此模块在Apache2.0中更具灵活性。现在，它可以更改 `mod_proxy` 使用的请求头信息，并可以有条件地设置应答头信息。

`mod_proxy`

代理模块已经被完全重写以充分利用新的过滤器结构的优势，从而实现一个更为可靠的HTTP/1.1代理模块。另外，新的 `<Proxy>` 配置段提供了更具可读性(而且更快)的代理站点控制；同时，重载 `<Directory "proxy:...">` 指令的方法已经不再被支持了。这个模块现在依照协议支持分为 `proxy_connect` 、 `proxy_ftp` 、 `proxy_http` 三个部分。

`mod_negotiation`

新的 `ForceLanguagePriority` 指令可以确保在所有情况下客户端都收到一个单一文档，以取代不可接受或多选择的响应。另外，内容协商和MultiViews算法已经进行了优化以提供更完美的结果，并提供了包括文档内容的新类型表。

`mod_autoindex`

经过自动索引后的目录列表现在可被配置为使用HTML表格从而使格式更加清晰，而且允许更为细化的排序控制，包括版本排序和通配符过滤目录列表。

`mod_include`

新的指令集允许修改默认的SSI元素的开始和结束标签，而且允许以主配置文件里的错误提示和时间格式的配置取代SSI文档中的相应部分。正则表达式(现在已基于Perl的正则表达式语法)的解析和分组结果可以用 `mod_include` 的变量 `$0` .. `$9` 取得。

`mod_auth_dbm`

现在可以使用 `AuthDBMType` 指令支持多种类似DBM的数据库。

The Apache License, Version 2.0

Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. **Definitions**

   "License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

   "Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. **Grant of Copyright License.** Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. **Grant of Patent License.** Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. **Redistribution.** You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

    i. You must give any other recipients of the Work or Derivative Works a copy of this License; and
    ii. You must cause any modified files to carry prominent notices stating that You changed the files; and

iii. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
iv. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. **Submission of Contributions.** Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. **Trademarks.** This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. **Disclaimer of Warranty.** Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. **Limitation of Liability.** In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. **Accepting Warranty or Additional Liability.** While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

参考手册

编译与安装

本文仅阐述Apache在Unix和类Unix系统中的编译和安装，在Windows中的编译和安装可以参考在Microsoft Windows中使用Apache。其他平台可以参见针对特定平台的说明文档。

像其它许多开源项目一样，Apache使用 `libtool` 和 `autoconf` 建立编译环境。

如果你只进行很小的版本升级(比如2.2.54→2.2.55)，请直接跳转到下面的升级部分。

针对心急者的概述

下载	`$ lynx http://httpd.apache.org/download.cgi`
解压	`$ gzip -d httpd-_NN_.tar.gz` `$ tar xvf httpd-_NN_.tar` `$ cd httpd-_NN_`
配置	`$ ./configure --prefix=_PREFIX_`
编译	`$ make`
安装	`$ make install`
配置	`$ vi _PREFIX_/conf/httpd.conf`
测试	`$ _PREFIX_/bin/apachectl -k start`

其中*NN*必须用当前的副版本号替代；*PREFIX*是服务器被安装到文件系统中的路径名，如果没有指定*PREFIX*，默认会装到 `/usr/local/apache2`

下面从编译和安装Apache httpd的要求开始详细阐述编译和安装的每个步骤。

要求

编译Apache的要求如下：

磁盘空间

必须保证有50MB以上的自由临时磁盘空间。Apache安装完毕后会占据10MB左右的空间，实际的磁盘空间需求会因编译设置和是否安装第三方模块而有所不同。

ANSI-C编译器及编译环境

必须装有ANSI-C编译器，推荐使用自由软件基金会(FSF)的GCC。如果没有GCC，那么要确保使用的编译器符合ANSI标准，而且 `PATH` 中必须包含指向基本编译工具比如 `make` 的路径。

确保准确的时间

由于HTTP协议的元素都会用到时间，有必要了解一下你的系统所使用的时间同步机制。在基于网络时间协议(NTP)的系统中，一般是用 `ntpdate` 或 `xntpd` 来同步时间。有关NTP软件的资料请参见NTP主页。

Perl 5 [可选]

有些用Perl写的支持脚本，如 `apxs` 或 `dbmmanage` ，需要Perl5解释器(5.003或以上的版本就足够了)。如果系统中存在多个Perl解释器，比如有系统提供的Perl 4，还有你自己安装的Perl 5，推荐你使用 `--with-perl` 选项来确保 `configure` 脚本使用正确的版本。如果 `configure` 没有没找到Perl 5也没关系，这并不影响Apache httpd的编译和安装，只是相关的支持脚本不能使用而已。

apr/apr-util >= 1.2

`apr` 和 `apr-util` 包含在Apache httpd的发行源代码中，并且在绝大多数情况下使用都不会出现问题。当然，如果 `apr` 或 `apr-util` 的1.0或1.1版本已经安装在你的系统中了，则必须将你的 `apr` / `apr-util` 升级到1.2版本，或者将httpd单独分开编译。要使用发行源代码中自带的 `apr` / `apr-util` 源代码进行安装，你必须手动完成：

```
# 编译和安装 apr 1.2
cd srclib/apr
./configure --prefix=/usr/local/apr-httpd/
make
make install
# 编译和安装 apr-util 1.2
cd ../apr-util
./configure --prefix=/usr/local/apr-util-httpd/ --with-apr=/usr/local/apr-httpd/
make
make install
# 配置 httpd
cd ../../
./configure --with-apr=/usr/local/apr-httpd/ --with-apr-util=/usr/local/apr-util-httpd/
```

下载

Apache可以从Apache HTTP服务器下载站点及其镜像站点下载。大多数类UNIX系统的Apache用户最好的选择是下载源代码并编译一个适合自己的版本，这个过程(下面将要讲述)是很简单的，它允许你根据自己的需求进行定制。另一方面编译好的二进制版本通常没有进

行及时的更新。如果你下载的是编译好的二进制版本，请按照其中的 `INSTALL.bindist` 文件进行安装。

下载完毕后，应该对下载来的tar包作PGP签名校验，以确保其完整而且未被篡改过。(PGP下载页面)

解压

解压Apache httpd的tar包很简单：

```
$ gzip -d httpd-_NN_.tar.gz

$ tar xvf httpd-_NN_.tar
```

这样就在当前目录下新建了一个包含发行版源代码的目录，必须 `cd` 进入这个目录以继续服务器的编译。

配置源代码树

这一步是根据你的特定平台和个人需求配置源代码树。位于发行源代码根目录的 `configure` 脚本会完成这个步骤(对下载Apache CVS版源代码的开发者，需要装有 `autoconf` 和 `libtool` ，并需要执行 `buildconf` ，对于官方的发行版则没这个必要)。

要想用所有的默认值配置源代码树只要简单的执行 `./configure` 命令就可以了，同时 `configure` 还可以接受命令行参数以改变默认值。

最重要的选项是Apache安装目录的前缀： `--prefix` ，因为Apache需要知道这个目录才能正常运作。更多的微调选项请参考配置选项详解。

这样，你就可以通过在配置选项中指定要启用或禁用哪些模块来定制Apache的特性。Base组的模块默认包含在Apache中。其他组的模块可以通过 `--enable-module` 指令启用。其中module是模块名去掉" `mod_` "并将下划线转换成连字符后的字符串。你也可以使用 `--enable-module=shared` 指令将模块编译为可在运行时加载和卸载的动态共享对象(DSO)。同样的，你也可以使用 `--disable-module` 指令禁用Base组的模块。注意，使用这些指令的时候 `configure` 不会对你拼写错误的模块发出警告说找不到某某模块，而只是简单的忽略这个选项。

另外，有时候还必须提供给 `configure` 脚本关于编译器、库、头文件位置的更多信息。这些可以通过环境变量或者命令行选项传递给 `configure` 脚本。要了解更多信息，请参考 `配置源代码树` 。

为了让你对能指定什么有一个简单的印象，此例演示编译Apache，并将其安装在 `/sw/pkg/apache` 目录，指定了一个特定的编译器以及编译参数，而且允许今后将两个附加的模块 `mod_rewrite` 和 `mod_speling` 通过DSO机制在运行时动态加载：

```
    $ CC="pgcc" CFLAGS="-O2" \
     ./configure --prefix=/sw/pkg/apache \
     --enable-rewrite=shared \
     --enable-speling=shared
```

`configure` 需要运行几分钟，以测试指定的功能在你的系统中是否有效，并建立稍后编译时所需的许多Makefile文件。

编译

运行以下命令你就可以编译Apache的各个部分了：

```
$ make
```

请耐心等候，因为对一个基本配置的编译，需要运行几分钟左右，实际需要的时间会因为你的硬件和选择的模块数量有很大不同。

安装

现在可以在*PREFIX*目录(参见上述的 `--prefix` 参数)下安装了，执行：

```
$ make install
```

如果是升级，安装程序不会覆盖你的配置文件和文档。

配置

接着，通过修改 `_PREFIX_/conf/` 目录下的配置文件，来配置Apache HTTP服务器。

```
$ vi _PREFIX_/conf/httpd.conf
```

docs/manual/下有Apache使用手册，http://httpd.apache.org/docs/2.2/有最新的文档，你还可以查看完整的指令索引。

测试

现在，可以执行下述命令立即启动你的Apache HTTP服务器：

```
$ _PREFIX_/bin/apachectl -k start
```

你应该可以用 `http://localhost/` 来请求你的第一个网页了，这个网页位于 `DocumentRoot` 目录下，通常是 `_PREFIX_/htdocs/` 。随后，可以这样停止服务器：

```
$ _PREFIX_/bin/apachectl -k stop
```

升级

升级的第一步是阅读源代码目录中的发布公告(release announcement)和 `CHANGES` 文件以寻找可能会对你的站点产生影响的变化。如果主板本号的变化(例如1.3→2.0或2.0→2.2)表明编译时和运行时的配置发生了重大变化，需要手动调整，所有模块也需要升级以兼容新版本的模块API 。

小幅度的版本升级(例如：2.2.55→2.2.57)很容易。 `make install` 的过程不会改写任何已经存在的文档、日志、配置文件。此外，开发者也会尽量兼容上一版本的 `configure` 选项、运行时配置、模块API 。大多数情况下，你将能够使用与上一版本完全相同的 `configure` 命令行和运行时配置，而你原来的所有模块也将正常工作。

如果你保存了上一次安装后 `build` 子目录中的 `config.nice` 文件，升级将更加平滑。这个文件精确地保存了所有对目录树进行配置的 `configure` 命令行。你只需要将 `config.nice` 文件复制到新的源代码目录树的根文件夹并进行你希望的修改后，然后运行下面的命令即可完成升级：

```
$ ./config.nice
$ make
$ make install
$ _PREFIX_/bin/apachectl -k graceful-stop
$ _PREFIX_/bin/apachectl -k start
```

你应该总是在将新版本的Apache投入正式运行前，对这个新版本进行足够的、针对你的实际运行环境的测试。比如，你可以使用一个不同的 `--prefix` 设置将新版本安装在一个不同的目录，并使用 `Listen` 指令在一个不同的端口监听。经过一段时间的测试以发现可能存在的问题，然后再做出最后的决定。

启动Apache

在Windows NT/2000/XP/2003操作系统中，Apache一般以服务方式运行，或者在Windows 95/98/ME中以控制台程序方式运行。详情请参见以服务方式运行Apache和以控制台程序方式运行Apache。

在Unix操作系统中， `httpd` 程序作为一个守护进程运行，在后台不断处理请求。本文档描述了如何调用 `httpd` 。

Apache是怎样启动的

如果配置文件中 `Listen` 定义的是默认的80端口(或1024以下)，那么启动Apache将需要root权限以将它绑定在特权端口上。一旦服务器开始启动并完成了一些诸如打开日志文件之类的准备操作，它将创建很多子进程来完成一些诸如侦听和回应客户端请求的工作。 `httpd` 主进程仍然以root用户的权限运行，而它的子进程将以一个较低权限的用户运行。这将由你选择的多路处理模块进行控制。

调用 `httpd` 可执行文件的推荐方法是使用 `apachectl` 控制脚本。此脚本设置了在某些操作系统中正常运行 `httpd` 所必需的环境变量，然后调用 `httpd` 二进制文件。 `apachectl` 会传递命令行的所有参数，因此所有用于 `httpd` 的选项多半也可以用于 `apachectl` 。你可以直接修改 `apachectl` 脚本，改变首部的 `HTTPD` 变量使之指向 `httpd` 可执行文件的正确位置，也可以设置任意的命令行参数，使之总是有效。

`httpd` 被调用后第一件要做的事情就是找到并读取配置文件 `httpd.conf` 。此文件的位置是在编译时设定的，但也可以象下面这样在运行时用 `-f` 选项来指定：

```
/usr/local/apache2/bin/apachectl -f /usr/local/apache2/conf/httpd.conf
```

如果启动过程一切正常，服务器将与终端分离并几乎立即出现命令行提示符。这表示服务器已经启动并开始运行。然后你就可以用你的浏览器去连接你的服务器来查看 `DocumentRoot` 目录下的测试文档及其页面链接里的其它文档的本地副本。

启动时发生错误

如果Apache在启动过程中发生了致命错误，它将在退出前把描述这个错误的信息显示在终端上或者写入到 `ErrorLog` 中。一个最常产生的错误信息是" `Unable to bind to Port ...` "，这主要由以下原因造成：

- 想由一个特权端口启动服务但没有以root用户运行
- 启动服务时已经有另外的Apache实例在运行或其他的web服务器已经绑定了同样的端口

更多问题的解决办法，请参见常见问题。

随系统启动时启动

如果你希望你的服务器在系统重启后仍保持运行状态，你应该把 `apachectl` 的调用加入到你的系统启动文件中(通常为 `rc.local` 文件或 `rc.N` 目录下的某一文件)。这将会以root权限启动Apache。当然，在此之前，你必须保证你的服务器已经完成了安全和访问权限的设定。

`apachectl` 脚本被设计为可以用作SysV初始化脚本，它接受 `start` 、 `restart` 、 `stop` 参数，并把它们翻译为 `httpd` 对应的信号，所以通常都可以将 `apachectl` 连接到适当的初始目录，但是需要检查你的系统对此的精确要求。

额外信息

关于 `httpd` 和 `apachectl` 以及其他相关支持程序的命令行选项的详细信息请参见服务器和支持程序页面。其中还包括所有的随Apache发行包发布的模块和它们提供的指令的文档。

停止和重启

本文档叙述了在类Unix系统上如何停止和重启Apache。 Windows NT/2000/XP/2003的用户请参见以服务方式运行Apache， Windows 9x/ME用户则参见在控制台中运行Apache。

简介

为了停止或者重新启动Apache，你必须向正在运行的 `httpd` 进程发送信号。有两种发送信号的方法。第一种方法是直接使用UNIX的 `kill` 命令向运行中的进程发送信号。你也许你会注意到你的系统里运行着很多 `httpd` 进程。但你不应该直接对它们中的任何一个发送信号，而只要对已经在 `PidFile` 中记载下了自身PID的父进程发送信号。也就是说，你不必对父进程以外的任何进程发送信号。你可以向父进程发送三种信号： `TERM` 、 `HUP` 、 `USR1` ，我们过一会儿再进行详细的说明。

你可以用下面这样的命令来向父进程发送信号：

```
kill -TERM `cat /usr/local/apache2/logs/httpd.pid`
```

第二种方法是使用下面将要描述的 `httpd` 二进制可执行文件的 `-k` 命令行选项： `stop` 、 `restart` 、 `graceful` 、 `graceful-stop` 。不过我们推荐你使用 `apachectl` 控制脚本来向 `httpd` 二进制可执行文件传递这些选项。

当你向 `httpd` 发送信号后，你可以这样来读取它的进行过程：

```
tail -f /usr/local/apache2/logs/error_log
```

你可以修改这些示例以适应你的 `ServerRoot` 和 `PidFile` 设置。

立即停止

信号：TERM

`apachectl -k stop`

发送 `TERM` 或 `stop` 信号到父进程可以使它立刻杀死所有子进程。这将花费一些时间来杀死所有子进程。然后父进程自己也退出。所有进行中的请求将被强行中止，而且不再接受其它请求。

优雅重启

信号：USR1

`apachectl -k graceful`

`USR1` 或 `graceful` 信号使得父进程建议子进程在完成它们现在的请求后退出(如果他们没有进行服务，将会立刻退出)。父进程重新读入配置文件并重新打开日志文件。每当一个子进程死掉，父进程立刻用新的配置文件产生一个新的子进程并立刻开始伺服新的请求。

重启代码的设计能够确保MPM进程控制指令的正常运作，也就是在重启过程中确保有适当数量的进程和线程以响应客户端的请求。它是这样 `StartServers` 的：如果在一秒钟以后还没有新创建 `StartServers` 个子进程，则创建出足够完成现在任务的子进程个数。因此，代码除了保有能够维持服务器的现有负载数量的子进程外，也确保 `StartServers` 按你的意愿运作。

使用 `mod_status` 的用户会注意到在 `USR1` 信号发出后，服务器的统计信息没有被清零。代码被写成既能将你服务器无法伺服新请求的时间降至最少(这些请求将被操作系统放到队列里，使得它们不会丢失)，又能遵从你的参数优化。为了做到这一点，它将在重新生成子进程的过程中，在*scoreboard*上保存所有子进程的状态。

`mod_status` 还会将那些在优雅重启前就已经开始而没有结束伺服请求的子进程用一个" `G` "来标志。

目前，日志滚动脚本还无法使用 `USR1` 来确定所有写入预重启日志的子进程都已结束。我们建议你在发出了 `USR1` 信号后等待一个适当的时间，然后再对旧的日志做处理。比如说如果对于一个窄带用户来说，大部分的点击处理将在10分钟之内完成，那么你应该在处理旧的日志前等待15分钟。

如果Apache重启时发现配置文件有误，那么父进程将不会重启，而是报错并退出。在优雅重启的情况下，它将在处理中的子进程存在的情况下维持它的存在(就是那些被要求在处理完它们的请求后"优雅退出"的子进程)。如果你要重启服务器，这将导致一些问题：它将不能绑定到它的监听端口。在执行重启之前，你可以用 `-t` 命令行参数来检查配置文件语法的正确性(参见 `httpd`)。但这仍然不能保证服务器一定可以正确的重启。为了从语法和语义两方面检查配置文件，你可以用一个非root用户来启动 `httpd` 。如果没有错误，它将尝试去打开套接字和日志文件，继而因没有root权限而失败(或是因为现在运行的 `httpd` 已经绑定了这些端口)。如果是因为其他原因那么就可能是一个配置文件产生的错误，你就应当在进行优雅重启之前改正这个错误。

立即重启

信号：HUP

`apachectl -k restart`

向父进程发送 `HUP` 或 `restart` 信号会使它象收到 `TERM` 信号一样杀掉所有的子进程，不同之处在于父进程本身并不退出。它重新读入配置文件、重新打开日志文件。然后产生一系列新的子进程来继续服务。

使用 `mod_status` 的用户会注意到在 `HUP` 信号发出后，服务器统计信息会被清零。

如果你重启时配置文件有误，那么父进程将不会重启，而是报错并退出。参见上文中避免的方法。

优雅停止

信号：WINCH

`apachectl -k graceful-stop`

`WINCH` 或 `graceful-stop` 信号使得父进程建议子进程在完成它们现在的请求后退出(如果他们没有进行服务，将会立刻退出)。然后父进程删除 `PidFile` 并停止在所有端口上的监听。父进程仍然继续运行并监视正在处理请求的子进程，一旦所有子进程完成任务并退出或者超过由 `GracefulShutdownTimeout` 指令规定的时间，父进程将会退出。在超时的情况下，所有子进程都将接收到 `TERM` 信号并被强制退出。

在"优雅"状态下， `TERM` 信号将会立即中止父进程和所有子进程。由于 `PidFile` 已经被删除，你将无法使用 `apachectl` 或 `httpd` 发送该信号。

```
graceful-stop允许你同时运行多个相同配置的[httpd](#calibre_link-54)实例。这在对Apache进行平滑升级的
必须注意确保诸如`Lockfile`和`ScriptSock`之类的磁盘文件包含服务器的PID ，并且能够安全的共存。然而如果一
你还必须防止潜在的竞争条件，比如使用`rotatelogs`风格的管道日志。运行中的多个`rotatelogs`实例企图同时滚
```

附录：信号和竞争条件

在Apache 1.2b9 之前，有很多关于重启和死亡信号的竞争条件。关于竞争条件的一个简单描述是：一个时间敏感的问题，如果一些事情在不适当的时间或以不恰当的顺序发生，它将作出你不期望的反应；如果同样的事情在恰当的时间发生，则不会出现异常。凭借那些拥有"正确"特性设置的体系结构，我们尽量避免了它们的出现。但值得注意的是，仍然有一些竞争条件存在于这样的体系结构中。

使用物理磁盘的 `ScoreBoardFile` 就有损坏ScoreBoard的潜在危险。这将发生在"bind: Address already in use"(`HUP` 之后)或"long lost child came home!"(`USR1` 之后)时。前者是一个致命错误，而后者则会使服务器丢失ScoreBoard的一个记录。所以我们建议多使用优雅重启，偶尔使用硬重启。这些问题很难解决，但幸运的是大多数结构并不需要ScoreBoard文件。而如果你需要这样的结构，你可以参考 `ScoreBoardFile` 文档。

当每个子进程在一个HTTP的持续连接(KeepAlive)中涉及到第二个并发的请求时，所有的结构都会或多或少存在竞争状态的问题。它将在读取了请求而没有读取任何请求头之后立刻退出。这个修复对于1.2来说来得太晚了。但因为持续连接的客户端已经考虑到网络延时和服务

器超时会造成类似的情况，所以理论上说，这不是一个太大的问题。而实际上似乎也没有任何影响：在一个测试案例中服务器在一秒之内被重启了20次，而客户端却成功的浏览了网站，而且没有任何破损的图片或空文档。

配置文件

本页阐述了Apache服务器的配置文件。

主配置文件

相关模块

- `mod_mime`

相关指令

- `<IfDefine>`
- `Include`
- `TypesConfig`

Apache的配置文件是包含若干指令的纯文本文件。主配置文件通常叫 `httpd.conf` ，其位置是编译时确定的，但可以用命令行参数 `-f` 来改变。另外，还可以用 `Include` 指令和通配符附加许多其他配置文件。任何配置文件都可以使用任何指令。只有在启动或重新启动Apache后，主配置文件的更改才会生效。

服务器还会读取一个包含MIME文件类型的文件，其文件名由 `TypesConfig` 指令确定，默认值是 `mime.types` 。

配置文件的语法

Apache配置文件的每一行包含一个指令，在行尾使用反斜杠"\"可以表示续行，但是反斜杠与下一行之间不能有任何其他字符(包括空白字符)。

配置文件中的指令是不区分大小写的，但是指令的参数(argument)通常是大小写敏感的。以"#"开头的行被视为注解并被忽略。注解不能出现在指令的后边。空白行和指令前的空白字符将被忽略，因此可以采用缩进以保持配置层次的清晰。

可以用 `apachectl configtest` 或者命令行选项 `-t` 检查配置文件中的错误，而无须启动Apache服务器。

模块

相关模块

- `mod_so`

相关指令

- `<IfModule>`
- `LoadModule`

Apache是模块化的服务器，这意味着核心中只包含实现最基本功能的模块。扩展功能可以作为模块动态加载。默认情况下，只有base组的模块被编译进了服务器。如果服务器在编译时包含了DSO模块，那么各模块可以独立编译，并可随时用 `LoadModule` 指令加载；否则，要增加或删除模块必须重新编译整个Apache。用于特定模块的指令可以用 `<IfModule>` 指令包含起来，使之有条件地生效。

用命令行参数 `-l` 可以查看已经编译到服务器中的模块。

指令的作用域

相关模块

- `<Directory>`
- `<DirectoryMatch>`
- `<Files>`
- `<FilesMatch>`
- `<Location>`
- `<LocationMatch>`
- `<VirtualHost>`

相关指令

主配置文件中的指令对整个服务器都有效。如果你只想改变某一部分的配置，你可以把指令嵌入
到 `<Directory>` 、 `<DirectoryMatch>` 、 `<Files>` 、 `<FilesMatch>` 、 `<Location>` 、 `<LocationMat`
配置段中，这样就可以限制指令的作用域为文件系统中的某些位置或特定的URL。这些配置段还可以进行嵌套，以进行更精细的配置。

Apache还具备同时支持多个站点的能力，称为虚拟主机。 `<VirtualHost>` 配置段中的指令仅对该段中的特定站点(虚拟主机)有效。

虽然大多数指令可以包含在任意的配置段中，但是某些指令仅在某些特定的范围内才有意义。比如，控制进程建立的指令仅在主服务器范围内有效。要查询一个指令可以被应用于哪些配置段中，可以查看该指令的作用域项。更详细资料可以查看配置段说明。

.htaccess 文件

相关模块

- `AccessFileName`
- `AllowOverride`

相关指令

Apache可以使用分布在整个网站文件目录树结构中的特殊文件来进行分散配置，这些特殊的文件通常叫 `.htaccess` ，但是也可以用 `AccessFileName` 指令来改变它的名字。 `.htaccess` 文件中指令的作用域是存放它的那个目录及其所有子目录。 `.htaccess` 文件的语法与主配置文件相同。由于对每次请求都会读取 `.htaccess` 文件，所以对这些文件的修改会立即生效。

要了解一个指令是否可以用在 `.htaccess` 文件中，可以查阅该指令的作用域项。服务器管理员可以在主配置文件中使用 `AllowOverride` 指令来决定哪些指令可以在 `.htaccess` 文件中生效。

有关 `.htaccess` 文件更详细的资料，可以查看.htaccess 指南。

配置段(容器)

配置文件中指令的作用范围可能是整个服务器，也可能是特定的目录、文件、主机、URL。本文阐述如何使用配置段(容器)以及 `.htaccess` 文件来改变配置指令的作用范围。

配置段(容器)的类型

相关模块

- `core`
- `mod_version`
- `mod_proxy`

相关指令

- `<Directory>`
- `<DirectoryMatch>`
- `<Files>`
- `<FilesMatch>`
- `<IfDefine>`
- `<IfModule>`
- `<IfVersion>`
- `<Location>`
- `<LocationMatch>`
- `<Proxy>`
- `<ProxyMatch>`
- `<VirtualHost>`

容器有两种基本类型。大多数容器是针对各个请求的，包含于其中的指令仅对与该容器匹配的请求起作用，而容器 `<IfDefine>` 、 `<IfModule>` 、 `<IfVersion>` 仅在启动和重新启动中起作用，如果在启动时指定的条件成立，则其中的指令对所有的请求都有效，否则将被忽略。

`<IfDefine>` 容器中的指令只有在 `httpd` 命令行中设定了特定的参数后才有效。下例中，只有在服务器用 `httpd -DClosedForNow` 方式启动时，所有的请求才会被重定向到另一个站点：

```
<IfDefine ClosedForNow>
Redirect / http://otherserver.example.com/
</IfDefine>
```

`<IfModule>` 容器很相似，但是其中的指令只有当服务器启用特定的模块时才有效(或是被静态地编译进了服务器，或是被动态装载进了服务器)，注意，配置文件中该模块的装载指令 `LoadModule` 行必须在出现在此容器之前。这个容器应该仅用于你希望无论特定模块是否安装，配置文件都能正常运转的场合；而不应该用于容器中的指令在任何情况下都必须生效的场合，因为它会抑制类似模块没找到之类的有用出错信息。

下例中， `MimeMagicFiles` 指令仅当 `mod_mime_magic` 模块启用时才有效。

```
<IfModule mod_mime_magic.c>

MimeMagicFile conf/magic

</IfModule>
```

`<IfVersion>` 指令与 `<IfDefine>` 和 `<IfModule>` 很相似，但是其中的指令只有当正在执行的服务器版本与指定的版本要求相符时才有效。这个模块被设计用于测试套件、以及在一个存在多个不同httpd版本的大型网络中需要分别针对不同版本使用不同配置的情况。

```
 <IfVersion >= 2.1>

# 仅在版本高于 2.1.0 的时候才生效
 </IfVersion>
```

`<IfDefine>` 、 `<IfModule>` 、 `<IfVersion>` 都可以在条件前加一个"!"以实现条件的否定，而且都可以嵌套以实现更复杂的配置。

文件系统和网络空间

最常用的配置段是针对文件系统和网络空间特定位置的配置段。首先必须理解文件系统和网络空间这两个概念的区别，文件系统是指操作系统所看见的磁盘视图，比如，在Unix文件系统中，Apache会被默认安装到 `/usr/local/apache2` ，在Windows文件系统中，Apache会被默认安装到 `"C:/Program Files/Apache Group/Apache2"` (注意：Apache始终用正斜杠而不是反斜杠作为路径的分隔符，即使是在Windows中)。相反，网络空间是网站被web服务器发送以及被客户在浏览器中所看到的视图。所以网络空间中的路径 `/dir/` 在Apache采用默认安装路径的情况下对应于Unix文件系统中的路径 `/usr/local/apache2/htdocs/dir/` 。由于网页可以从数据库或其他地方动态生成，因此，网络空间无须直接映射到文件系统。

文件系统容器

`<Directory>` 和 `<Files>` 指令与其相应的正则表达式版本(`<DirectoryMatch>` 和 `<FilesMatch>`)一起作用于文件系统的特定部分。 `<Directory>` 配置段中的指令作用于指定的文件系统目录及其所有子目录，.htaccess文件可以达到同样的效果。下例中， `/var/web/dir1` 及其所有子目录被允许进行目录索引。

```
<Directory /var/web/dir1>
Options +Indexes
</Directory>
```

`<Files>` 配置段中的指令作用于特定的文件名，而无论这个文件实际存在于哪个目录。下例中的配置指令如果出现在配置文件的主服务器段，则会拒绝对位于任何目录下的 `private.html` 的访问。

```
<Files private.html>
Order allow,deny
Deny from all
</Files>
```

`<Files>` 和 `<Directory>` 段的组合可以作用于文件系统中的特定文件。下例中的配置会拒绝对 `/var/web/dir1/private.html` 、 `/var/web/dir1/subdir2/private.html` 、 `/var/web/dir1/subdir3/private.html` 等任何 `/var/web/dir1/` 目录下 `private.html` 的访问。

```
<Directory /var/web/dir1>
<Files private.html>
Order allow,deny
Deny from all
</Files>
</Directory>
```

网络空间容器

`<Location>` 指令与其相应的正则表达式版本(`<LocationMatch>`)一起作用于网络空间的特定部分。下例中的配置会拒绝对任何以" `/private` "开头的URL路径的访问，比如： `http://yoursite.example.com/private` 、 `http://yoursite.example.com/private123` 、 `http:` 等所有以" `/private` "开头的URL路径。

```
<Location /private>
Order Allow,Deny
Deny from all
</Location>
```

`<Location>` 指令与文件系统无关，下例演示了如何将特定的URL映射到Apache内部的处理器 `mod_status` ，而并不要求文件系统中确实存在 `server-status` 文件。

```
<Location /server-status>

SetHandler server-status

</Location>
```

通配符和正则表达式

`<Directory>` 、 `<Files>` 、 `<Location>` 指令可以使用与C标准库中的 `fnmatch` 类似的shell风格的通配符。"*"匹配任何字符串，"?"匹配任何单个的字符，"[*seq*]"匹配*seq*序列中的任何字符，符号"/"不被任何通配符所匹配，所以必须显式地使用。

如果需要更复杂的匹配，这些容器都有一个对应的正则版本： `<DirectoryMatch>` 、 `<FilesMatch>` 、 `<LocationMatch>` ，可以使用与Perl兼容的正则表达式，以提供更复杂的匹配。但是还必须注意下面配置段的合并部分关于使用正则表达式会如何作用于配置指令的内容。

下例使用非正则表达式的通配符来改变所有用户目录的配置：

```
<Directory /home/*/public_html>

Options Indexes

</Directory>
```

下例使用正则表达式一次性拒绝对多种图形文件的访问：

```
<FilesMatch \.(?i:gif|jpe?g|png)$>

Order allow,deny

Deny from all

</FilesMatch>
```

什么情况下用什么

选择使用文件系统容器还是使用网络空间容器其实很简单。当指令应该作用于文件系统时，总是用 `<Directory>` 或 `<Files>` ；而当指令作用于不存在于文件系统的对象时，就用 `<Location>` ，比如一个由数据库生成的网页。

绝对不要试图用 `<Location>` 去限制对文件系统中的对象的访问，因为许多不同的网络空间路径可能会映射到同一个文件系统目录，从而导致你的访问限制被突破。比如：

```
<Location /dir/>

Order allow,deny

Deny from all

</Location>
```

上述配置对 `http://yoursite.example.com/dir/` 请求的确起作用。但是设想在一个不区分大小写的文件系统中，这个访问限制会被 `http://yoursite.example.com/DIR/` 请求轻易突破。而 `<Directory>` 指令才会真正作用于对这个位置的任何形式的请求。但是有一个例外，就是Unix文件系统中的符号连接(软连接)，符号连接可以使同一个目录出现在文件系统中的多个位置。 `<Directory>` 指令将不重设路径名而直接追踪符号连接，因此，对于安全要求最高的，应该用 `Options` 指令禁止对符号连接的追踪。

不要认为使用大小写敏感的文件系统就无所谓了，因为有很多方法可以将不同的网络空间路径映射到同一个文件系统路径，所以，应当尽可能使用文件系统容器。但是也有一个例外，就是把访问限制放在 `<Location />` 配置段中可以很安全地作用于除了某些特定URL以外的所有URL。

虚拟主机

`<VirtualHost>` 容器作用于特定的虚拟主机，为同一个机器上具有不同配置的多个主机提供支持。详见虚拟主机文档。

代理

`<Proxy>` 和 `<ProxyMatch>` 容器中的指令仅作用于通过 `mod_proxy` 代理服务器访问的、与指定URL匹配的站点。下例中的配置会拒绝通过代理服务器访问 `cnn.com` 站点。

```
<Proxy http://cnn.com/*>

Order allow,deny

Deny from all

</Proxy>
```

允许使用哪些指令？

查阅指令的作用域，就可以知道哪些指令可以出现在哪些配置段中。从语法上看，允许在 `<Directory>` 段中使用的指令当然也可以在 `<DirectoryMatch>` 、 `<Files>` 、 `<FilesMatch>` 、 `<Location>` 、 `<LocationMatch>` 、 `<Proxy>` 、 `<ProxyMatch>` 段中使用，但也有例外：

- `AllowOverride` 指令只能出现在 `<Directory>` 段中。
- `Options` 中的 `FollowSymLinks` 和 `SymLinksIfOwnerMatch` 只能出现在 `<Directory>` 段或者 `.htaccess` 文件中。
- `Options` 指令不能用于 `<Files>` 和 `<FilesMatch>` 段。

配置段的合并

配置段会按非常特别的顺序依次生效，由于这会对配置指令的处理结果产生重大影响，因此理解它的流程非常重要。

合并的顺序是：

1. `<Directory>` (除了正则表达式)和 `.htaccess` 同时处理；(如果允许的话， `.htaccess` 的设置会覆盖 `<Directory>` 的设置)
2. `<DirectoryMatch>` (和 `<Directory ~>`)
3. `<Files>` 和 `<FilesMatch>` 同时处理
4. `<Location>` 和 `<LocationMatch>` 同时处理

除了 `<Directory>` ，每个组都按它们在配置文件中出现的顺序被依次处理，而 `<Directory>` (上面的第1组)，会按字典顺序由短到长被依次处理。例如： `<Directory /var/web/dir>` 会先于 `<Directory /var/web/dir/subdir>` 被处理。如果有多个指向同一个目录的 `<Directory>` 段，则按它们在配置文件中的顺序被依次处理。用 `Include` 指令包含进来的配置被视为按原样插入到 `Include` 指令的位置。

位于 `<VirtualHost>` 容器中的配置段在外部对应的段处理完毕以后再处理，这样就允许虚拟主机覆盖主服务器的设置。

当请求是由 `mod_proxy` 处理的时候， `<Proxy>` 容器将会在处理顺序中取代 `<Directory>` 容器的位置。

后面的段覆盖前面的相应的段。

技术说明

其实，在名称翻译阶段(即用 `Aliases` 和 `DocumentRoots` 来映射URL到文件系统)之前，会有一个 `<Location>` / `<LocationMatch>` 的序列被处理，而在名称翻译结束后，这个序列的处理结果则被完全抛弃。

一些例子

这是一个演示合并顺序的例子。如果这些指令都起作用，则会按 A > B > C > D >E 的顺序依次生效。

```
<Location />
E
</Location>
<Files f.html>
D
</Files>
<VirtualHost *>
<Directory /a/b>
B
</Directory>
</VirtualHost>
<DirectoryMatch "^.*b$">
C
</DirectoryMatch>
<Directory /a/b>
A
</Directory>
```

在下面这个更具体的例子中，无论在 `<Directory>` 段中加了多少访问限制，由于 `<Location>` 段将会被最后处理，从而会允许不加限制的对服务器的访问，可见合并的顺序是很重要的，千万小心！

```
<Location />
Order deny,allow
Allow from all
</Location>
# 这个<Directory>段将不会实际生效
<Directory />
Order allow,deny
Allow from all
Deny from badguy.example.com
</Directory>
```

缓冲指南

这篇文档是对 `mod_cache` 、 `mod_disk_cache` 、 `mod_mem_cache` 、 `mod_file_cache` 和 htcacheclean参考文档内容的增补。它描述了如何利用Apache的缓冲特性来加速web和代理(proxy)服务，同时避免一些常见的问题和配置错误。

简介

从Apache2.2起， `mod_cache` 和 `mod_file_cache` 将不再是试验模块，它们已经足够稳定，可以用于实际生产中了。这些缓冲体系提供了一个强有力的途径来加速原始web服务器(origin webserver)和代理服务器(proxy)的HTTP处理速度。

`mod_cache` 以及它的支持模块 `mod_mem_cache` 和 `mod_disk_cache` 提供了智能的HTTP缓冲。内容(content)本身被存储在缓冲区中， `mod_cache` 的目的在于管理控制内容缓冲能力的各种HTTP头和选项。它可以同时处理本地的内容和代理的内容。 `mod_cache` 被设计为同时针对简单的和复杂的缓冲配置，以用于处理代理的内容、动态的本地内容、必须加速访问的随时间而改变本地文件。

另一方面， `mod_file_cache` 实现了一个更基本的、但是在某些情况下更有效的缓冲形式，它避免了主动确保URL缓冲能力所需的维护复杂性， `mod_file_cache` 通过提供文件句柄(file-handle)和内存映射(memory-mapping)的技巧来维持一个自Apache最后一次启动以来的文件缓冲区。同样地， `mod_file_cache` 的目标是改善不常修改的本地静态文件的访问时间。

由于 `mod_file_cache` 提供了一个相对简单的缓冲实现，除了 `CacheFile` 和 `MMapStatic` 文档的特定段落之外，这篇指南的说明覆盖了 `mod_cache` 的缓存体系结构。

为了更好的理解这篇文档，你应当熟悉HTTP的基础知识，并且已经阅读过从URL到文件系统的映射和内容协商这两篇用户指南。

缓冲概述

相关模块

- `mod_cache`
- `mod_mem_cache`
- `mod_disk_cache`
- `mod_file_cache`

相关指令

- `CacheEnable`

- `CacheDisable`
- `MMapStatic`
- `CacheFile`
- `CacheFile`
- `UseCanonicalName`
- `CacheNegotiatedDocs`

在一个请求的生存期中， `mod_cache` 内可能会发生两个主要阶段。首先， `mod_cache` 将是一个URL映射模块，也就是说，如果一个URL已经被缓存并且这个缓存尚未失效，该请求将由 `mod_cache` 直接处理。

这也意味着在处理一个请求时通常还要发生的其他阶段：比如由 `mod_proxy` 或 `mod_rewrite` 处理的阶段，将不会发生。不过，这正是将内容缓存起来的关键所在。[虽然某些阶段被省略了，但这正是启用缓冲特性的初衷：减少处理步骤以提高速度。]

如果这个URL不在缓存中， `mod_cache` 将会在请求的处理过程中添加一个过滤器。在Apache使用通常的方法定位内容之后，该过滤器将会在内容被用于服务以后运行。如果该内容被确定为可以缓存，那么它将被保存在缓冲区中以便为将来的请求提供服务。

如果该URL存在于缓存中并且已经失效的话，该过滤器同样会被添加，但是 `mod_cache` 将会同时向后端(backend)提交一个条件请求以确定缓存的版本是否是当前的最新版本。如果是最新版本，那么它的元信息(meta-information)将会被更新并且使用这个缓存的版本来服务于该请求。如果不是最新版本，那么过滤器将使用刚才为请求提供服务的最新内容更新缓存。

提高缓存命中率

在缓存本地生成的内容的时候，将 `UseCanonicalName` 指令设置为 `On` 可以显著提高缓存的命中率。这是由于为缓冲区提供内容的虚拟主机的主机名是缓冲键(cache key)的组成部分。当该指令设置为 `On` 时，具有多个服务器名或别名的虚拟主机将不会产生不同的缓存实体，而是按照各自的规范主机名(canonical hostname)来存储。

由于缓存发生在将URL映射到文件系统的阶段，缓存的文档将只被用来响应对URL的请求。通常情况下这没什么重大意义，但是当你使用服务端包含(Server Side Includes)时，这一点将显得特别重要：

```
<!-- 下面的包含可以被缓存 -->
<!--#include virtual="/footer.html" -->

<!-- 下面的包含不可以被缓存 -->
<!--#include file="/path/to/footer.html" -->
```

如果你使用服务端包含(SSI)，并且希望从缓冲中获得快速服务好处，你应当使用 `virtual` 类型的包含。

失效周期(Expiry Periods)

缓存实体的默认失效周期是一个小时(3600秒)，当然这个可以轻易的通过`CacheDefaultExpire`指令来修改。这个默认值仅仅用在产生内容的原始资源没有明确指定失效时间或最后修改时间的情况下。

如果一个应答没有包含`Expires`头但却包含`Last-Modified`头时，`mod_cache`可以根据`CacheLastModifiedFactor`指令推断出失效周期。

对于本地内容，`mod_expires`可以用来调整失效周期。

失效周期的最大值还可以通过`CacheMaxExpire`指令来控制。

关于条件请求(Conditional Request)的简短说明

当缓存的内容失效并且被从后端(backend)或内容提供者(content provider)那里重新请求的时候，Apache并不直接传递原始的请求，而是使用一个条件请求(conditional request)。

HTTP协议使用的一些头(header)允许客户端或缓冲区鉴别同一个内容的不同版本。例如，如果一个资源应答了"Etag:"头，那么就可以生成一个包含"If-Match:"头的条件请求；如果一个资源应答了"Last-Modified:"头，那么就可以生成一个包含"If-Modified-Since:"头的条件请求；等等。

对于这样的条件请求，应答的不同取决于内容是否匹配这些条件。如果一个请求包含一个"If-Modified-Since:"头，而请求的内容在指定的时间之后并未发生改变，那么一个简洁的"304 Not Modified"应答就可以了。

如果请求的内容已经变化，那么将按照原来没有条件请求的普通方式来应答。

和缓存相关的条件请求的好处有两个方面。首先，当向后端提交这样的条件请求时，如果从后端获得的内容与存储的内容相匹配(这很容易确定)，就可以避免由于传递全部资源所带来的开销。

其次，条件请求通常只让后端花费较小的开销。对于静态文件，通常所有的开销就是一个`stat()`或类似的系统调用以确定改文件的大小是否变化以及最后修改时间。这样，如果被请求的内容尚未变化，甚至在Apache缓存的本地内容已经失效的情况下，仍然可以从缓冲区中快速取得以服务于请求——只要从缓冲区读取比从后端读取更快(例如从内存缓冲区读取就比从硬盘上读取更快)。

什么可以被缓存？

如前所述，Apache中的缓冲存在两种不同工作方式。`mod_file_cache`的缓冲区负责维护Apache启动时的文件内容。当一个存在于该模块缓冲区中的文件被请求时，该请求将被拦截并用缓冲区中的内容为其提供服务。

`mod_cache` 的缓冲区相对而言较为复杂。当服务于一个请求时，如果它先前并未被缓存，则缓冲模块将会判断该内容是否可以被缓存。判断应答的缓冲能力(cachability)基于以下条件：

1. 必须为该URL启用了缓冲。请参考 `CacheEnable` 和 `CacheDisable` 指令。
2. 应答必须具有如下HTTP状态码：200, 203, 300, 301, 410 。
3. 该请求必须是一个HTTP GET请求。
4. 如果请求包含"Authorization:"头，则应答不被缓存。
5. 如果应答包含"Authorization:"头，它必须同时也在"Cache-Control:"头中包含"s-maxage"、"must-revalidate"或"public"选项。
6. 如果该URL包含一个请求字符串(比如来自于一个使用GET方法的HTML表格)，除非应答包含"Expires:"头，否则将不被缓存。这是基于RFC2616的13.9小节的规范。
7. 如果应答的状态码是200(OK)，除非明确打开了 `CacheIgnoreNoLastMod` 指令，否则该应答还必须至少包含一个"Etag"、"Last-Modified"或"Expires"头才能被缓存。
8. 如果应答头"Cache-Control:"中包含"private"选项，除非明确打开了 `CacheStorePrivate` 指令，否则将不被缓存。
9. 同样，如果应答头"Cache-Control:"中包含"no-store"选项，除非明确打开了 `CacheStoreNoStore` 指令，否则将不被缓存。
10. 如果应答包含"Vary:"头，并且其中包含通配符"*"(匹配所有)，也将不被缓存。

什么不应该被缓存？

简而言之，随时间变化的内容不应该被缓存；取决于特定请求的内容不应该被缓存；依赖于不被HTTP内容协商处理的请求的内容也不应该被缓存。[本句翻译的很没把握，原文：In short, any content which is highly time-sensitive, or which varies depending on the particulars of the request that are not covered by HTTP negotiation, should not be cached.]

如果你有某些动态内容，它们的变化依赖于请求发起者的IP地址或者差不多每5分钟就会发生改变，那么这些内容毫无疑问是不应该被缓存的。

另一方面，如果内容的变化依赖于各种HTTP头，更加明智的做法可能是通过使用"Vary"头进行缓存。

易变的/协商的内容

当 `mod_cache` 接收到一个后端发出的、带有"Vary"头的应答的时候，它将尽可能智能的处理它。如果有机会， `mod_cache` 将会检查之后进入的请求的"Vary"头属性，然后用正确的缓冲区内容为这个请求提供服务。

举个例子来说，接收到一个带有如下"Vary"头的应答：

```
Vary: negotiate,accept-language,accept-charset
```

`mod_cache` 将只会使用与原始请求的accept-language和accept-charset头匹配的缓存内容来提供服务。

安全方面的考虑

授权(Authorisation)，访问控制(Access & Control)

`mod_cache` 非常像一个内置的反向代理(reverse-proxy)。除非必须要向后端提交请求，否则缓冲模块将直接为请求提供服务。对于缓冲本地资源，这种模式彻底改变了Apache的安全模型。

因为遍历文件系统的目录结构以寻找可能存在的 `.htaccess` 文件是一个开销非常昂贵的操作，它部分地抵消了缓冲所带来的好处(加速请求)，所以 `mod_cache` 并不检查缓存中的实体是否被允许(authorised)用于提供服务。换句话说，只要 `mod_cache` 中缓存的内容尚未失效，那么它们将被直接用于为请求提供服务。

举例来说，如果你为某个资源按照IP地址配置了访问许可，你必须要确保这些内容不被缓存。你可以使用 `CacheDisable` 指令或 `mod_expires` 模块达到这个目的。不做权限检查的 `mod_cache` 模块非常像一个反向代理：缓存内容并用缓存的内容向任意IP地址上的任意客户提供服务。

本地利用(Local exploits)

因为终端用户的请求可以由缓冲区直接提供服务，所以缓冲区自身便成为一个那些企图干扰、破坏内容的攻击者的攻击目标。很重要的、需要牢记的一点是：缓冲区必须始终对运行Apache的用户是可写的。这正好与通常的原则：始终保持所有内容对运行Apache的用户不可写，完全相反！

如果运行Apache的用户是潜在的不安全用户，比如，通过一个有漏洞的CGI进程，就有可能对缓冲区发起攻击，当使用 `mod_disk_cache` 的时候，就很容易插入或者修改缓冲区中内容。

这样一来，运行Apache的用户就会增加一个与其它类型的攻击相比更加危险的安全隐患。如果你正在使用 `mod_disk_cache` ，你必须时刻牢记：确保为Apache及时打上所有的安全补丁并且使用suEXEC以一个不同于运行Apache用户的其他用户身份运行CGI进程。

缓存中毒(Cache Poisoning)

当将Apache作为一个缓冲代理服务器运行的时候，将可能存在一个所谓"缓存中毒"的问题。"缓存中毒"是一个泛称术语，用于指代各种造成代理服务器从后台检索到错误内容的攻击。

举个例子来说，如果你运行Apache的系统所使用的DNS服务器发生了DNS缓存中毒，攻击者将可能欺骗Apache连接到一个错误的服务器去请求内容。另一个例子是所谓的HTTP请求走私(request-smuggling)攻击。

这篇文档并不是深入探讨HTTP请求走私的地方(你应当去问google)，但有一点你必须知道：攻击者可以通过制造一连串的请求并利用原始web服务器的漏洞，达到完全控制代理服务器所检索到的内容的目的。

文件句柄缓冲(File-Handle Caching)

相关模块

- `mod_file_cache`
- `mod_mem_cache`

相关指令

- `CacheFile`
- `CacheEnable`
- `CacheDisable`

打开文件的动作本身就是一个造成延时的原因，特别是打开网络文件系统中的文件。通过维护一个保存高使用率文件的文件描述符的缓冲区，Apache就可以避免这种延时。当前，Apache提供了两种不同的文件句柄缓冲实现方法。

缓冲文件(CacheFile)

存在于Apache中最基本的缓冲方式是由 `mod_file_cache` 实现的文件句柄(file-handle)缓冲。胜于缓存文件内容本身，这个缓冲区维护一张打开的文件描述符表，用于保存在配置文件中使用 `CacheFile` 指令指定的文件的文件句柄。

`CacheFile` 指令指示Apache在启动时打开某个文件并且为所有之后对这个文件的访问重复使用这个文件句柄。

```
CacheFile /usr/local/apache2/htdocs/index.html
```

如果你打算使用这种方式缓存大量的文件句柄，你必须确保操作系统对同时打开的文件的数量限制是足够的。

虽然使用 `CacheFile` 不会导致文件的内容被缓存，但是将会导致在Apache运行期间所有对文件的更改都不会生效。用于提供服务的文件的内容将从Apache启动以来一直保持不变。

如果在Apache运行期间文件被删除了，Apache将会持续维护一个打开的文件描述符并且使用Apache启动时文件的内容来提供服务。这个通常也意味着虽然文件已经被删除，并且不在文件系统中显示出来，但是释放的空间并不会被覆盖，直到Apache被停止、文件描述符被关闭。

CacheEnable fd

`mod_mem_cache` 也提供了一个文件句柄缓冲方案，可以通过 `CacheEnable` 指令来启用。

```
CacheEnable fd /
```

与 `mod_cache` 的方案相比，这种方案更加智能：缓存内容失效以后相应的句柄将不再被维护。

内存缓冲(In-Memory Caching)

相关模块

- `mod_mem_cache`
- `mod_file_cache`

相关指令

- `CacheEnable`
- `CacheDisable`
- `MMapStatic`

直接从系统的内存中提供服务通常是取得服务内容最快速的方法。从一个磁盘控制器读取文件，或者更糟糕的是从远程网络读取文件，其速度要慢上几个数量级。磁盘控制器通常涉及到物理动作，访问网络要受限于网络带宽，而访问内存通常仅仅只需要几毫微秒时间。

内存也许是目前单位字节最昂贵的存储器，保证它充分发挥作用非常重要。将文件缓存在内存中将导致系统可用内存的减少。正如我们将要看到的，在操作系统存在内存缓冲区的情况下，这不是一个大问题。但是当使用Apache自己的内存缓冲区的情况下，确保没有为缓冲区分配太多的内存就显得十分重要。否则，操作系统将会使用swap(虚拟内存/交换区)，这可能会导致性能急剧下降。

操作系统缓冲

几乎所有现代的操作系统都由内核直接管理文件数据在内存中的缓冲。这是一个强有力的特性，并且在极大程度上操作系统做的非常好。比如在Linux系统上，让我们看看第一次读取一个文件和第二次读取同样的文件所需要的时间：

```
colm@coroebus:~$ time cat testfile > /dev/null
real    0m0.065s
user    0m0.000s
sys     0m0.001s
colm@coroebus:~$ time cat testfile > /dev/null
real    0m0.003s
user    0m0.003s
sys     0m0.000s
```

即使对于这样的一个小文件，两次读取的时间差异也十分惊人。这是由于内核在内存中缓存了文件的内容。

通过确保在你的系统上始终存在"多余的"内存，你就可以确保会有越来越多的文件内容被缓存在这个缓冲区中。这是一个非常有效的内存缓冲途径，并且根本无需对Apache作出任何额外的配置。

另外，由于操作系统知道文件何时被修改或删除了，它就可以自动的从内存缓冲区中删除失效的文件内容。这是一个优于Apache自身的内存缓冲区的巨大优点，因为Apache无法得知文件被修改或删除的信息。

尽管操作系统自动管理的缓冲区有着性能和洞悉文件状态的优势，但是在某些情况下Apache自己的内存缓冲却更加有效。

首先，操作系统只能缓存它自己知道的文件，如果你将Apache当作一个代理服务器运行，那么Apache可以缓存非本地文件。如果你还想要无可匹敌的内存缓存速度，也必须使用Apache自己的内存缓冲区。

MMapStatic 缓冲

`mod_file_cache` 提供了 `MMapStatic` 指令，它可以指示Apache在启动时将一个静态文件的内容映射到内存中(使用mmap()系统调用)。Apache将会使用内存中缓存的内容来为后来对这个文件的访问提供内容。

```
MMapStatic /usr/local/apache2/htdocs/index.html
```

使用了 `CacheFile` 指令以后，在Apache运行期间，对这些文件所做的任何修改都不会生效。

`MMapStatic` 指令并不关心它占用了多少内存，所以你必须确保不要过度滥用这个指令。每个Apache子进程都将复制这部分内存，所以非常重要的一点是你必须确保被映射的文件不能占用太多的内存，以至于操作系统不得不使用交换区或虚拟内存。

mod_mem_cache 缓冲

`mod_mem_cache` 提供了一个智能的HTTP内存缓冲方案。它同时也直接使用堆内存，这也意味着即使MMap不被你的操作系统所支持， `mod_mem_cache` 仍然能够实现缓冲。

这种类型的缓冲可以通过以下方法启用：

```
# 启用内存缓冲
CacheEnable mem /

# 将缓冲区的大小限制为 1 MB
MCacheSize 1024
```

磁盘缓冲(Disk-based Caching)

相关模块

- `mod_disk_cache`

相关指令

- `CacheEnable`
- `CacheDisable`

`mod_disk_cache` 为 `mod_cache` 提供了一个基于磁盘的缓冲机制。和 `mod_mem_cache` 一样，这是一种智能缓冲，仅在缓存内容没有失效的情况下才从缓冲区中提供服务。

通常，这个模块将按如下方式进行配置：

```
CacheRoot   /var/cache/apache/
CacheEnable disk /
CacheDirLevels 2
CacheDirLength 1
```

请注意，因为缓冲区位于本地磁盘上，所以操作系统的内存缓冲区通常对它们的访问也有效。所以虽然这些文件被存储在本地磁盘上，但若这些文件被频繁的访问，那么很可能操作系统已经将它们保存在内存中了。

深入理解缓冲存储区(Cache-Store)

要将项目保存在缓冲区中，`mod_disk_cache` 会为被请求的URL创建一个22字符的哈希值。该哈希值包含了该URL的主机名、协议、端口、路径、CGI变量，以确保多个URL不会发生碰撞。

这22个字符的取值范围是64个不同的字符，这意味着最多可以有22^64种可能的取值。例如，一个URL的哈希值可能是：`xyTGxSMO2b68mBCykqkp1w` 。这个哈希值将被用作缓存中对应于那个URL的文件名前缀，但是首先，这个哈希值将被按照 `CacheDirLevels` 和 `CacheDirLength` 指令分解成每一级目录名。

`CacheDirLevels` 指定了子目录的层数，`CacheDirLength` 指定了每级子目录名的字符数。使用上述例子的设置，这个哈希值将被转化成如下文件名前缀：`/var/cache/apache/x/y/TGxSMO2b68mBCykqkp1w` 。

使用这种技术的总体目标是减少某个特定目录中子目录或文件的个数，因为绝大多的文件系统在子目录或文件数过多的情况下的访问速度都会大打折扣。将 `CacheDirLength` 设置为"1"将使得任意一层目录下的子目录数都不会超过64，若为设为"2"则为64*64，依此类推。除非你有一个非常好的理由，否则"1"将是 `CacheDirLength` 指令的推荐值。

如何设置 `CacheDirLevels` 指令的值取决于你预计到将会在缓冲区中保存多少个文件。上述示例使用的"2"将会导致大约会有4096个子目录最终被建立，大约100万个文件被缓存，大约平均每个文件夹存储245个URL缓冲文件。

每个URL在缓冲区中至少会使用两个文件。通常，一个是包含了URL元信息(meta-information)的".header"文件，比如何时失效；另一个是".data"文件，包含了按字节复制的用于为URL提供服务的内容。

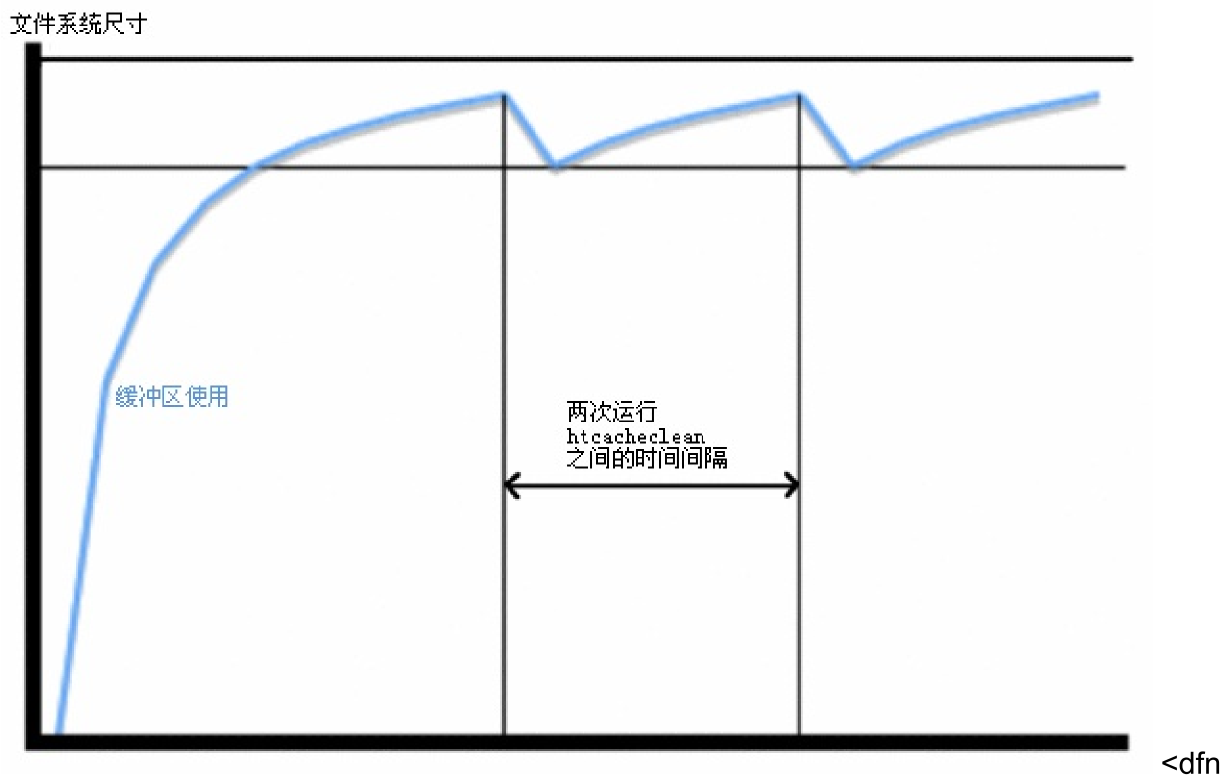
在通过使用"Vary"头进行内容协商的情况下，将会为该URL创建一个".vary"目录，该目录下将会保存多个适合不同协商内容的".data"文件。

维护磁盘缓冲区

虽然 `mod_disk_cache` 将会删除缓冲区中失效的文件，但是它并不负责维护整个缓冲区总共究竟应该占据多大空间以及至少要保留多少剩余空间。

作为弥补，Apache附带了一个htcacheclean工具，正如你从它的名字猜到的，它可以周期性的清理缓冲区。确定htcacheclean的运行频率以及缓冲区应当占有多大的磁盘空间是一件复杂的事情。必须要经过多次尝试和碰壁才能找到一个最佳值。

htcacheclean有两种运作模型。一种是作为后台守护进程运行，或者由cron周期性的调用。htcacheclean经常使用一个小时或更多的时间来处理非常巨大的(几十G)缓冲区，所以如果你是使用cron来调用它的话，建议你测试一下多长时间运行一次比较合适，以避免在同一时间运行多个实例。



<dfn class="calibre50">图1</dfn>: 一个典型的缓冲区增长和清理的周期

因为 `mod_disk_cache` 模块自身并不关心究竟实际使用了多少磁盘空间，所以你必须确保htcacheclean被配置为在清理了缓冲区以后预留了足够多的"增长空间"。

服务器全局配置

本文档说明了由服务器核心(`core`)提供的以实现基本服务器运转的一些指令。

服务器标识

相关模块

- `ServerName`
- `ServerAdmin`
- `ServerSignature`
- `ServerTokens`
- `UseCanonicalName`
- `UseCanonicalPhysicalPort`

相关指令

`ServerAdmin` 和 `ServerTokens` 指令控制有关服务器的哪些信息将出现在服务器生成的文档中(如错误消息)。 `ServerTokens` 指令设置服务器HTTP响应头字段的值。

`ServerName` 、 `UseCanonicalName` 、 `UseCanonicalPhysicalPort` 指令用来决定怎样构建自引用URL，譬如，某客户端对一个目录发出请求，但没有包含目录名最后的斜线"/"，Apache将重定向客户端到包含"/"的全名，以使得客户端可以正确解析文档中的相对引用。

文件定位

相关模块

- `CoreDumpDirectory`
- `DocumentRoot`
- `ErrorLog`
- `LockFile`
- `PidFile`
- `ScoreBoardFile`
- `ServerRoot`

相关指令

这些指令控制Apache正常工作所需的各种文件的定位。如果路径名不以斜线(/)开头，那么就认为该文件是相对于 `ServerRoot` 的相对路径，需要注意路径中的文件哪些对非root用户来说是可写的，参见安全提示以获得更多细节。

限制资源的使用

相关模块

- `LimitRequestBody`
- `LimitRequestFields`
- `LimitRequestFieldsize`
- `LimitRequestLine`
- `RLimitCPU`
- `RLimitMEM`
- `RLimitNPROC`
- `ThreadStackSize`

相关指令

`LimitRequest*` 系列指令用来限制Apache在读取客户端请求的过程中使用的资源数量。通过限制这些值，可以减轻某些拒绝服务(DOS)攻击。

`RLimit*` 系列指令限制被Apache子进程所派生的进程使用的资源数量，通常，这些指令用来控制CGI脚本和SSI exec命令所使用的资源。

`ThreadStackSize` 指令在某些平台上用来控制堆栈大小。

日志文件

要有效地管理Web服务器，就有必要反馈服务器的活动、性能以及出现的问题。Apache HTTP服务器提供了非常全面而灵活的日志记录功能。本文将阐述如何配置文件以及如何理解日志内容。

安全警告

任何人只要对Apache存放日志文件的目录具有写权限，也就当然地可以获得启动Apache的用户(通常是root)的权限，绝对不要随意给予任何人存放日志文件目录的写权限。细节请参见安全方面的提示。

另外，日志文件可能会包含未加转换的来自用户的信息，用户就有机会恶意插入控制符，所以处理原始日志时应该当心这个问题。

错误日志(Error Log)

相关模块

- `ErrorLog`
- `LogLevel`

相关指令

错误日志是最重要的日志文件，其文件名和位置取决于 `ErrorLog` 指令。Apache httpd将在这个文件中存放诊断信息和处理请求中出现的错误，由于这里经常包含了出错细节以及如何解决，如果服务器启动或运行中有问题，首先就应该查看这个错误日志。

错误日志通常被写入一个文件(unix系统上一般是 `error_log` ，Windows和OS/2上一般是 `error.log`)。在unix系统中，错误日志还可能被重定向到 `syslog` 或通过管道操作传递给一个程序。

错误日志的格式相对灵活，并可以附加文字描述。某些信息会出现在绝大多数记录中，一个典型的例子是：

```
[Wed Oct 11 14:32:52 2000] [error] [client 127.0.0.1]
client denied by server configuration:
/export/home/live/ap/htdocs/test
```

其中，第一项是错误发生的日期和时间；第二项是错误的严重性， `LogLevel` 指令使只有高于指定严重性级别的错误才会被记录；第三项是导致错误的IP地址；此后是信息本身，在此例中，服务器拒绝了这个客户的访问。服务器在记录被访问文件时，用的是文件系统路径，而不是Web路径。

错误日志中会包含类似上述例子的多种类型的信息。此外，CGI脚本中任何输出到 `stderr` 的信息会作为调试信息原封不动地记录到错误日志中。

用户可以增加或删除错误日志的项。但是对某些特殊请求，在访问日志(access log)中也会有相应的记录，比如上述例子在访问日志中也会有相应的记录，其状态码是403，因为访问日志也可以定制，所以可以从访问日志中得到错误事件的更多信息。

在测试中，对任何问题持续监视错误日志是非常有用的。在unix系统中，可以这样做：

```
tail -f error_log
```

访问日志(Access Log)

相关模块

- `mod_log_config`
- `mod_setenvif`

相关指令

- `CustomLog`
- `LogFormat`
- `SetEnvIf`

访问日志中会记录服务器所处理的所有请求，其文件名和位置取决于 `CustomLog` 指令， `LogFormat` 指令可以简化日志的内容。这里阐述如何配置服务器的访问日志。

实施日志管理，首先当然必须产生访问日志，然后才能分析日志从而得到有用的统计信息。日志分析不是Web服务器的职责，已超出本文的范畴，更多资料和有关分析工具的信息，可以查看Open Directory或Yahoo 。

不同版本的Apache httpd使用了不同的模块和指令来控制对访问的记录，包括mod_log_referer, mod_log_agent和 `TransferLog` 指令。现在， `CustomLog` 指令包含了旧版本中相关指令的所有功能。

访问日志的格式是高度灵活的，使用很象C风格的printf()函数的格式字符串。下面有几个例子，完整的说明可以查看用于 `mod_log_config` 模块的格式字符串。

通用日志格式(Common Log Format)

这是一个典型的记录格式：

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common

 CustomLog logs/access_log common
```

它定义了一种特定的记录格式字符串，并给它起了个别名叫 `common` ，其中的"%"指示服务器用某种信息替换，其他字符则不作替换。引号(`"`)必须加反斜杠转义，以避免被解释为字符串的结束。格式字符串还可以包含特殊的控制符，如换行符" `\n` "、制表符" `\t` "。

`CustomLog` 指令建立一个使用指定别名的新日志文件，除非其文件名是以斜杠开头的绝对路径，否则其路径就是相对于 `ServerRoot` 的相对路径。

上述配置是一种被称为通用日志格式(CLF)的记录格式，它被许多不同的Web服务器所采用，并被许多日志分析程序所识别，它产生的记录形如：

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET
/apache_pb.gif HTTP/1.0" 200 2326
```

记录的各部分说明如下：

`127.0.0.1` (`%h`)

这是发送请求到服务器的客户的IP地址。如果 `HostnameLookups` 设为 `On` ，则服务器会尝试解析这个IP地址的主机名并替换此处的IP地址，但并不推荐这样做，因为它会显著拖慢服务器，最好是用一个日志后续处理器来判断主机名，比如 `logresolve` 。如果客户和服务器之间存在代理，那么记录中的这个IP地址就是那个代理的IP地址，而不是客户机的真实IP地址。

`-` (`%l`)

这是由客户端 `identd` 进程判断的RFC1413身份(identity)，输出中的符号"-"表示此处的信息无效。除非在严格控制的内部网络中，此信息通常很不可靠，不应该被使用。只有在将 `IdentityCheck` 指令设为 `On` 时，Apache才会试图得到这项信息。

`frank` (`%u`)

这是HTTP认证系统得到的访问该网页的客户标识(userid)，环境变量 `REMOTE_USER` 会被设为该值并提供给CGI脚本。如果状态码是401，表示客户未通过认证，则此值没有意义。如果网页没有设置密码保护，则此项将是" `-` "。

`[10/Oct/2000:13:55:36 -0700]` (`%t`)

这是服务器完成请求处理时的时间，其格式是：

```
[日/月/年:时:分:秒 时区] 日 = 2数字 月 = 3字母 年 = 4数字 时 = 2数字 分 = 2数字 秒 = 2数字 时区 = (+
```

可以在格式字符串中使用 `%{format}t` 来改变时间的输出形式，其中的 `format` 与C标准库中的 `strftime()` 用法相同。

`"GET /apache_pb.gif HTTP/1.0"` (`\"%r\"`)

引号中是客户端发出的包含许多有用信息的请求行。可以看出，该客户的动作是 `GET` ，请求的资源是 `/apache_pb.gif` ，使用的协议是 `HTTP/1.0` 。另外，还可以记录其他信息，如：格式字符串" `%m %U%q %H` "会记录动作、路径、查询字符串、协议，其输出和" `%r` "一样。

`200` (`%>s`)

这是服务器返回给客户端的状态码。这个信息非常有价值，因为它指示了请求的结果，或者是被成功响应了(以2开头)，或者被重定向了(以3开头)，或者出错了(以4开头)，或者产生了服务器端错误(以5开头)。完整的状态码列表参见HTTP规范(RFC2616第10章)。

`2326` (`%b`)

最后这项是返回给客户端的不包括响应头的字节数。如果没有信息返回，则此项应该是" `-` "，如果希望记录为" `0` "的形式，就应该用 `%B` 。

组合日志格式(Combined Log Format)

另一种常用的记录格式是组合日志格式，形式如下：

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{User-agent}i\"" combined

 CustomLog log/access_log combined
```

这种格式与通用日志格式类似，但是多了两个 `%{_header_}i` 项，其中的*header*可以是任何请求头。这种格式的记录形如：

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET
/apache_pb.gif HTTP/1.0" 200 2326
"http://www.example.com/start.html" "Mozilla/4.08 [en]
(Win98; I ;Nav)"
```

其中，多出来的项是：

`"http://www.example.com/start.html"` (`\"%{Referer}i\"`)

"Referer"请求头。此项指明了该请求是被从哪个网页提交过来的，这个网页应该包含有 `/apache_pb.gif` 或者其连接。

`"Mozilla/4.08 [en] (Win98; I ;Nav)"` (`\"%{User-agent}i\"`)

"User-Agent"请求头。此项是客户端提供的浏览器识别信息。

多文件访问日志

可以简单地在配置文件中用多个 `CustomLog` 指令来建立多文件访问日志。如下例，既记录基本的CLF信息，又记录提交网页和浏览器的信息，最后两行 `CustomLog` 示范了如何模拟 `ReferLog` 和 `AgentLog` 指令的效果。

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
CustomLog logs/referer_log "%{Referer}i -> %U"
CustomLog logs/agent_log "%{User-agent}i"
```

此例也说明了，记录格式可以直接由 `CustomLog` 指定，而并不一定要用 `LogFormat` 起一个别名。

条件日志

许多时候，根据与请求特征相关的环境变量来有选择地记录某些客户端请求会带来便利。首先，需要使用 `SetEnvIf` 指令来设置特定的环境变量以标识符合某种特定条件的请求，然后用 `CustomLog` 指令的 `env=` 子句，根据这些环境变量来决定记录或排除特定的请求。例如：

```
# 不记录本机发出的请求
SetEnvIf Remote_Addr "127\.0\.0\.1" dontlog
# 不记录对robots.txt文件的请求
SetEnvIf Request_URI "^/robots\.txt$" dontlog
# 记录其他请求
CustomLog logs/access_log common env=!dontlog
```

再如，将使用英语的请求记录到一个日志，而记录非英语的请求到另一个日志：

```
SetEnvIf Accept-Language "en" english
CustomLog logs/english_log common env=english
CustomLog logs/non_english_log common env=!english
```

虽然上述已经展示了条件日志记录的强大和灵活，但这不是控制日志内容的唯一手段，还可以用日志后继处理程序来剔除你不关心的内容，从而使日志更加有用。

日志滚动

即使一个并不繁忙的服务器，其日志文件的信息量也会很大，一般每10000个请求，访问日志就会增加1MB或更多。这就有必要定期滚动日志文件。由于Apache会保持日志文件的打开，并持续写入信息，因此服务器运行期间不能执行滚动操作。移动或者删除日志文件以后，必

须重新启动服务器才能让它打开新的日志文件。

用优雅的*(graceful)*方法重新启动，可以使服务器启用新的日志文件，而不丢失原来尚未写入的信息。为此，有必要等待一段时间，让服务器完成正在处理的请求，并将记录写入到原来的日志文件。以下是一个典型的日志滚动和为节省存储空间而压缩旧日志的例子：

```
mv access_log access_log.old
mv error_log error_log.old
apachectl graceful
sleep 600
gzip access_log.old error_log.old
```

另一种执行滚动的方法是使用下一节阐述的管道日志。

管道日志

Apache httpd可以通过管道将访问记录和出错信息传递给另一个进程，而不是写入一个文件，由于无须对主服务器进行编程，这个功能显著地增强了日志的灵活性。只要用管道操作符"`|`"后面跟一个可执行文件名，就可以使这个程序从标准输入设备获得事件记录。Apache在启动时，会同时启动这个管道日志进程，并且在运行过程中，如果这个进程崩溃了，会重新启动这个进程(所以我们称这个技术为"可靠管道日志")。

管道日志进程由其父进程Apache httpd产生，并继承其权限，这意味着管道进程通常是作为root运行的，所以保持这个程序简单而安全极为重要。

管道日志的一种重要用途是，允许日志滚动而无须重新启动服务器。为此，服务器提供了一个简单的程序 `rotatelogs` 。每24小时滚动一次日志的例子如下：

```
CustomLog "|/usr/local/apache/bin/rotatelogs /var/log/access_log 86400" common
```

注意：引号用于界定整个管道命令行。虽然这是针对访问日志的，但是其用法对于其他日志也一样。

在其他站点，有一个类似但更灵活的日志滚动程序叫cronolog 。

如果有较简单的离线处理日志的方案，就不应该使用条件日志和管道日志，即使它们非常强大。

虚拟主机

如果服务器配有若干虚拟主机，那么还有几个控制日志文件的功能。首先，可以把日志指令放在 `<VirtualHost>` 段之外，让它们与主服务器使用同一个访问日志和错误日志来记录所有的请求和错误，但是这样就不能方便的获得每个虚拟主机的信息了。

如果把 `CustomLog` 或 `ErrorLog` 指令放在 `<VirtualHost>` 段内，所有对这个虚拟主机的请求和错误信息会被记录在其私有的日志文件中，那些没有在 `<VirtualHost>` 段内使用日志指令的虚拟主机将仍然和主服务器使用同一个日志。这种方法对虚拟主机较少的服务器很有用，但虚拟主机非常多时，就会带来管理上的困难，还经常会产生文件描述符短缺的问题。

对于访问日志，有一个很好的折衷方案，在同一个访问日志文件中记录对所有主机的访问，而每条记录都注明虚拟主机的信息，日后再把记录拆开存入不同的文件。例如：

```
LogFormat "%v %l %u %t \"%r\" %>s %b" comonvhost

CustomLog logs/access_log comonvhost
```

`%v` 用来附加虚拟主机的信息。有个split-logfile程序可以根据不同的虚拟主机信息对日志进行拆分，并将结果存入不同的文件。

其他日志文件

相关模块

- `mod_logio`
- `mod_log_forensic`
- `mod_cgi`
- `mod_rewrite`

相关指令

- `LogFormat`
- `ForensicLog`
- `PidFile`
- `RewriteLog`
- `RewriteLogLevel`
- `ScriptLog`
- `ScriptLogBuffer`
- `ScriptLogLength`

记录接收和发送的实际字节数

`mod_logio` 增加了两个额外的 `LogFormat` 字段(%I 和 %O)用于记录接收和发送的实际字节数。

对比记录(Forensic Logging)

`mod_log_forensic` 提供了对客户端请求的对比记录，也就是在请求被处理之前和处理完成之后进行两次记录，所以对比日志(forensic log)对于每个请求都包含两条记录。对比记录器(forensic logger)十分严格，不可以进行定制。它可以成为无价的调试和安全工具。

PID文件

在启动时，Apache httpd将会在 `logs/httpd.pid` 文件中保存其父进程httpd的进程ID(process id[PID])。该文件名可以用 `PidFile` 指令改变。该PID可以被管理员利用来重新启动或者终止服务器后台守护进程。在Windows中，可以使用命令行参数 -k 。更多信息请参见停止和重新启动。

脚本日志

为了方便调试，可以用 `ScriptLog` 指令来记录CGI脚本的输入和输出。此功能应该仅用于测试，而不应该用于正常工作的服务器。更多资料请参见mod_cgi文档。

重写日志

在使用强大且灵活的mod_rewrite时，几乎都有必要用 `RewriteLog` 来帮助调试。这个日志提供了重写引擎如何转换请求的详细分解信息，其详细程度取决于 `RewriteLogLevel` 指令。

从URL到文件系统的映射

本文阐述Apache如何根据URL地址定位到文件在文件系统中的位置。

相关模块和指令

相关模块

- `mod_alias`
- `mod_proxy`
- `mod_rewrite`
- `mod_userdir`
- `mod_speling`
- `mod_vhost_alias`

相关指令

- `Alias`
- `AliasMatch`
- `CheckSpelling`
- `DocumentRoot`
- `ErrorDocument`
- `Options`
- `ProxyPass`
- `ProxyPassReverse`
- `ProxyPassReverseCookieDomain`
- `ProxyPassReverseCookiePath`
- `Redirect`
- `RedirectMatch`
- `RewriteCond`
- `RewriteMatch`
- `ScriptAlias`
- `ScriptAliasMatch`
- `UserDir`

DocumentRoot

Apache根据请求定位文件的默认操作是：取出URL路径(即URL中主机名和端口后面的部分)附加到由 `DocumentRoot` 指定的文件系统路径后面。这样就组成了在网上所看见的基本文件树结构。

如果服务器有多个虚拟主机，则Apache会使用下述两种方法之一：使用每个虚拟主机自己的 `DocumentRoot` 来组成文件系统路径，或者使用由 `mod_vhost_alias` 提供的指令基于IP地址或主机名动态地定位文件。

DocumentRoot以外的文件

实际应用中，经常有必要允许网络对 `DocumentRoot` 以外的文件进行访问。对此，Apache提供了多种方法，在Unix系统中，可以在文件系统的 `DocumentRoot` 目录下放置符号连接以访问其外部文件，考虑到安全问题，这种方法仅在相应目录的 `Options` 指令中设置了 `FollowSymLinks` 或 `SymLinksIfOwnerMatch` 时才有效。

另外，使用 `Alias` 指令可以将文件系统的任何部分映射到网络空间中。例如，这个命令

```
Alias /docs /var/web
```

可以把URL `http://www.example.com/docs/dir/file.html` 映射为 `/var/web/dir/file.html` 。 `ScriptAlias` 指令功能相似，而且使所有目标路径下的所有文件被视为CGI脚本。

`AliasMatch` 和 `ScriptAliasMatch` 指令可以实现基于正则表达式的匹配和替换，以提供更大的灵活性。例如：

```
ScriptAliasMatch ^/~([a-zA-Z0-9]+)/cgi-bin/(.+) /home/$1/cgi-bin/$2
```

上述命令可以将 `http://example.com/~user/cgi-bin/script.cgi` 映射到 `/home/user/cgi-bin/script.cgi` ，并视之为CGI脚本。

用户目录

在Unix系统中，一个特定用户"*user*"的主目录通常是" `~user/` "模块 `mod_userdir` 在网络上沿用了这个概念，允许使用URL访问位于各用户主目录下的文件，例如：

```
http://www.example.com/~user/file.html
```

出于安全原因，不应该给予网络用户直接操作主目录的权限，而应该在用户主目录下新建一个目录，把网络文件放在这个新建的目录中，并用 `UserDir` 指令告诉服务器。缺省的用户目录设置是" `Userdir public_html` "，因此，上述例子中的URL会映射

到 `/home/user/public_html/file.html` ，其中 `/home/user/` 是 `/etc/passwd` 指定的用户主目录。

当 `/etc/passwd` 没有指定主目录，那就要用到 `Userdir` 指令的另几种形式。

有些人觉得符号"~"(时常会被编码为 `%7e`)很别扭，希望用其他形式来表达用户目录。虽然模块 `mod_userdir` 并不支持，但是，如果合理规划服务器上的用户目录，则还是有可能用 `AliasMatch` 指令来达到这个目的。例如，如果希望将 `http://www.example.com/upages/user/file.html` 映射到 `/home/user/public_html/file.html` ，可以这样使用 `AliasMatch` 指令：

```
AliasMatch ^/upages/([a-zA-Z0-9]+)/?(.*) /home/$1/public_html/$2
```

URL重定向

上述指令都指示Apache返回给客户文件系统的某个特定内容，但是有时候，需要通知客户其请求的内容位于其他URL，并使客户产生新的对其他URL的请求，这种机制称为重定向*(redirection)*，可以用 `Redirect` 指令实现。例如：如果 `DocumentRoot` 的目录 `/foo/` 被转移到了 `/bar/` ，则可以这样引导客户访问新的位置：

```
Redirect permanent /foo/ http://www.example.com/bar/
```

这个命令重定向任何以 `/foo/` 开头的URL路径到位于同一个服务器 `www.example.com` 的 `/bar/` 。当然，可以重定向到任何其它服务器，而不仅仅是原来的那个。

Apache还提供了 `RedirectMatch` 指令来解决复杂的重定向问题。例如，要重定向对站点主页的请求到其他站点，而保留其他所有请求，可以这样配置：

```
RedirectMatch permanent ^/$ http://www.example.com/startpage.html
```

另一种方法是，暂时地重定向站点的所有页面到一个特定页面，如：

```
RedirectMatch temp .*  http://othersite.example.com/startpage.html
```

反向代理

Apache还允许将远程文档纳入本地服务器的网络空间中，因为Web服务器扮演一个代理服务器的角色(从远程服务器取得文档并返回给客户)，所以这种机制被称为反向代理*(reverse proxying)*，不同于标准代理的是，在客户看来，他请求的文档似乎原本就位于这个反向代理服务器上。

下例演示了当客户请求位于 `/foo/` 目录下的文档时，服务器从 `internal.example.com` 的 `/bar/` 目录下取回文档并返回给客户，似乎文档原本就在本地服务器上：

```
ProxyPass /foo/ http://internal.example.com/bar/

ProxyPassReverse /foo/ http://internal.example.com/bar/
ProxyPassReverseCookieDomain internal.example.com public.example.com
ProxyPassReverseCookiePath /foo/ /bar/
```

`ProxyPass` 指令使服务器正确地取回文档，同时， `ProxyPassReverse` 指令改变了起始于 `internal.example.com` 的请求，使之指向本地服务器上的目录。同样， `ProxyPassReverseCookieDomain` 和 `ProxyPassReverseCookieDomain` 指令将会改变后端服务器设置的cookie 。

需要注意的很重要的一点是，被取回的文档中的连接是不会被改写的，因此，文档中的所有绝对路径连接会突破代理机制而直接从 `internal.example.com` 取得。一个第三方模块mod_proxy_html可以用于重写HTML和XHTML连接。

URL重写引擎

`mod_rewrite` 模块提供了更强大的URL重写引擎，可以根据请求中诸如浏览器类型、源IP地址等特征来决定最终提交给客户的内容，还可以使用外部数据库或程序来决定如何处理一个请求，并可以执行上述的所有三种映射：内部重定向(aliases)、外部重定向、代理。许多实用程序都用到了这个模块，详细论述参见：URL重写指南。

File Not Found

从URL到文件系统的匹配失败是不可避免的，其产生原因有多种。有时是文档被转移了，对此最好是用URL重定向来引导用户访问新的位置，这样，虽然资源已经转移到新的位置，但是原来的书签和连接仍然有效。

另一种常见的原因是浏览器地址栏或者HTML连接中的URL被拼写错了，Apache提供了 `mod_speling` 模块来帮助解决这个问题，它会接管"File Not Found"错误并查找相似文件，如果找到了唯一的一个，则会重定向到这个文件，如果不止一个，则会列一张表反馈给用户。

`mod_speling` 的一个很有用的特性是，它可以忽略大小写查找文件，对不注意URL大小写的用户和unix文件系统尤为实用。但是，纠正偶然的URL错误会给服务器带来额外的负担，因为每次"不正确"的请求都将引发URL重定向和来自客户的新请求。

如果所有的努力都失败了，Apache会返回一个出错信息页面，其状态码为"404"(文件没找到)，其页面内容取决于 `ErrorDocument` 指令，并可以灵活地自定义其形式，详见：自定义错误响应。

安全方面的提示

本文中的提示和技巧有些是针对网络服务器的建立的，有些是综合性的，其余的则是针对Apache的。

保持不断更新和升级

Apache HTTP服务器有一个很好的安全记录和一个高度关注安全问题的开发社团。但是这仍然不能避免在发行版中存在或大或小的问题。所以知道这个软件的版本更新和升级补丁是至关重要的。如果你是直接从Apache组织得到Apache HTTP服务器的，我们强烈建议你订阅Apache HTTP服务器通告邮件列表以保证能够在第一时间得知软件的版本更新和升级补丁。许多第三方Apache软件发行版也有类似的服务。

当然，Web服务器出现的问题在绝大多数时候不是由Apache源代码引起的，而是由附加的代码、CGI脚本、底层操作系统引起的。因此你必须保持机器上所有软件的及时更新。

ServerRoot目录的权限

通常，Apache由root用户启动，在提供服务时切换为由 `User` 指令所指定的用户。正如root所执行的任何命令那样，你必须保证 `ServerRoot` 下的文件是受保护的，不允许非root用户对它修改。不仅文件本身，而且目录及其父目录都必须只能由root来改写。例如，如果将 `ServerRoot` 指定为 `/usr/local/apache` ，则推荐以root身份来建立此目录，如：

```
mkdir /usr/local/apache
cd /usr/local/apache
mkdir bin conf logs
chown 0 . bin conf logs
chgrp 0 . bin conf logs
chmod 755 . bin conf logs
```

这里已经假定了"/"、"/usr"、"/usr/local"只能由root来改写。在安装 `httpd` 可执行文件时，应该确保它也受到了同样的保护：

```
cp httpd /usr/local/apache/bin
chown 0 /usr/local/apache/bin/httpd
chgrp 0 /usr/local/apache/bin/httpd
chmod 511 /usr/local/apache/bin/httpd
```

你可以在其中建立htdocs子目录，该子目录可以允许其他用户改写 -- root不会执行其中任何文件，也不应该在其中建立文件。

如果允许非root用户对由root执行或读写的文件有写权限，则会危及系统。比如，别人有可能会覆盖 `httpd` 可执行文件，那么下一次启动时，就会执行恶意代码。如果日志目录(对非root用户)是可写的，别人就有可能用一个指向其他敏感文件的连接来覆盖日志文件，使那个文件被改写为杂乱的数据。如果日志文件本身(对非root用户)是可写的，别人就可能伪造日志。

服务器端包含

服务器端包含(SSI)会带来一些潜在的安全隐患。

首先是增加了服务器的负载。Apache必须解析所有允许SSI的文件，而无论其中是否包含SSI指令。虽然增加的负载较小，但是在共享服务器环境中会变得很显著。

SSI文件与CGI脚本一样存在风险。使用"exe ccmd"元素，允许SSI的文件可以执行任何CGI脚本，以及由httpd.conf设置的执行Apache的用户或组所允许执行的任何程序。

有若干方法可以在得到SSI好处的同时提高SSI文件的安全性。

服务器管理员可以使用关于CGI中所描述的suexec，以隔离野蛮SSI文件所造成的破坏。

对.html或.htm后缀的文件允许SSI是危险的，尤其是在一个共享的或者高流量的服务器环境中。被允许SSI的文件应该有一个单独的后缀，比如常规的.shtml，使服务器的负载保持在最低水平，并使风险管理更容易。

另一个方案是，关闭SSI页面执行脚本和程序的功能，即在用 `Options` 指令中，用 `IncludesNOEXEC` 替换 `Includes` 。注意，用户仍然可以使用 <--#include virtual="..." -->来执行位于 `ScriptAlias` 指令指定的目录中的CGI脚本。

关于CGI

首先，你不得不信任CGI程序的作者以及你自己发现CGI中潜在安全漏洞的能力，无论这些漏洞是有预谋的或者仅仅是意外。CGI脚本可以执行web服务器用户所允许执行的任意系统命令，如果没有经过仔细的检查，这可能是极其危险的。

由于所有CGI脚本都以相同的身份执行，所以可能会和其他脚本(有意或无意地)冲突。比如，用户A憎恨用户B，因此他就可能写一个脚本去破坏用户B的数据库。suEXEC是一个允许脚本以不同的身份运行的程序，它包含在Apache1.2以后的版本中，并被Apache服务器代码中特殊的挂钩所调用。还有一种常用的方法是使用CGIWrap 。

未指定为脚本的CGI

仅在下列情况下，可以考虑允许用户执行位于任意目录中的CGI脚本：

- 你绝对信任用户不会写一些有意无意会使系统遭受攻击的脚本。
- 你认为安全因素与其他因素相比显得不那么重要，存在一两个潜在漏洞也无关紧要。
- 你没有用户，而且没人会来访问你的服务器。

指定为脚本的CGI

把CGI集中在特定的目录中，并由管理员决定其中的内容。这样绝对比使用不作为脚本的CGI来得安全，除非对这些目录有写权限的用户被信任，或者管理员希望对每个CGI脚本/程序进行潜在安全漏洞测试。

大多数站点都选择这种方案，而不使用未指定为脚本的CGI。

其他动态内容的来源

嵌入在Apache中作为模块运行的脚本解释器，比如：mod_php, mod_perl, mod_tcl, mod_python 将会使用和Apache一样的用户身份运行(参见 `User` 指令)，所以被这些模块执行的脚本可能访问任何Apache服务器能够访问的对象。一些脚本引擎可能提供了某些方面的限制，但是最好在假定他们并不存在的前提下做好安全防护。

系统设置的保护

为了得到真正严密的保护，应该禁止用户使用可能导致安全特性被覆盖的 `.htaccess` 文件，方法是在服务器配置文件中设置：

```
<Directory />

  AllowOverride None

</Directory>
```

使所有目录无法使用 `.htaccess` 文件，明确指定可以使用的目录除外。

默认配置下服务器文件的保护

默认访问是偶尔会被误解的Apache特性之一。也就是，除非你采取措施，否则，如果服务器能够通过标准URL映射规则找到一个文件，那么就可能把它提供给客户端。比如下例：

```
# cd /; ln -s / public_html

Accessing http://localhost/~root/
```

它会允许客户端遍历整个文件系统。其解决方法是，在服务器配置中增加下列指令：

```
<Directory />
Order Deny,Allow
Deny from all
</Directory>
```

这样，对文件系统的默认访问被禁止。而对需要访问的区域，可以增加正确的 `Directory` 块，比如：

```
<Directory /usr/users/*/public_html>
  Order Deny,Allow
  Allow from all
</Directory>
<Directory /usr/local/httpd>
  Order Deny,Allow
  Allow from all
</Directory>
```

必须特别注意 `Location` 和 `Directory` 指令的相互作用，比如，即使 `<Directory />` 拒绝访问， `<Location />` 指令仍然可能推翻其设置。

还必须留意 `UserDir` 指令，此设置如果类似"./"，则与上述例子有相同的风险。如果你使用的是1.3或更高版本，我们强烈建议在服务器配置文件中包含以下指令：

```
UserDir disabled root
```

观察日志文件

要了解服务器上发生了什么，就必须检查日志文件。虽然日志文件只是记录已经发生的事件，但是它会让你知道服务器遭受的攻击，并帮助你判断是否提达到了必要的安全等级。

一些例子：

```
grep -c "/jsp/source.jsp?/jsp/ /jsp/source.jsp??" access_log
grep "client denied" error_log | tail -n 10
```

上例会列出试图使用Apache Tomcat Source.JSP Malformed Request Information Disclosure Vulnerability的攻击次数。下例会列出最后十个被拒绝的客户端：

```
[Thu Jul 11 17:18:39 2002] [error] [client foo.bar.com] client denied
by server configuration: /usr/local/apache/htdocs/.htpasswd
```

可见，日志文件只是记录已经发生的事件，所以，如果客户端可以访问 `.htpasswd` 文件，而且在访问日志中发现类似如下的记录：

```
foo.bar.com - - [12/Jul/2002:01:59:13 +0200] "GET /.htpasswd HTTP/1.1"
```

这可能表示服务器配置文件中的下列指令已经被注解了：

```
<Files ~ "^\.ht">

  Order allow,deny

  Deny from all

</Files>
```

动态共享对象(DSO)支持

Apache HTTP服务器是一个模块化的软件，管理员可以通过选择服务器中包含的模块进行功能增减。模块可以在编译时被静态包含进 `httpd` 二进制文件，也可以编译成独立于 `httpd` 二进制文件的动态共享对象(DSO)。DSO模块可以与服务器一起编译，也可以用Apache扩展工具(`apxs`)单独编译。

本文阐述如何使用DSO模块及其工作原理。

实现

相关模块

- `mod_so`

相关指令

- `LoadModule`

Apache对独立模块的DSO支持是建立在只能被静态编译进Apache核心的 `mod_so` 基础之上的，这是 `core` 以外唯一不能作为DSO存在的模块，而其他所有已发布的Apache模块，都可以通过安装文档中阐述中的 `编译选项` `--enable-_module_=shared` 被独立地编译成DSO并使之生效。一个被编译为 `mod_foo.so` 的DSO模块，可以在 `httpd.conf` 中使用 `mod_so` 的 `LoadModule` 指令，在服务器启动或重新启动时被加载。

新提供的支持程序 `apxs` (<dfn class="calibre27">APache eXtenSion</dfn>)可以在Apache源代码树之外编译基于DSO的模块，从而简化了Apache DSO模块的建立过程。其原理很简单：安装Apache时， `configure` 的 `make install` 命令会安装Apache C头文件，并把依赖于特定平台的编译器和连接器参数传给 `apxs` 程序，使用户可以脱离Apache的发布源代码树编译其模块源代码，而不改变支持DSO的编译器和连接器的参数。

用法概要

Apache2.0 的DSO功能简要说明：

1. 编译并安装已发布的Apache模块，比如编译 `mod_foo.c` 为 `mod_foo.so` 的DSO模块：

   ```
   $ ./configure --prefix=/path/to/install --enable-foo=shared
   $ make install
   ```

2. 编译并安装第三方模块，比如编译 `mod_foo.c` 为 `mod_foo.so` 的DSO模块：

```
$ ./configure --add-module=module_type:/path/to/3rdparty/mod_foo.c  --enable-foo=shar
$ make install
```

3. 配置Apache以便以后安装共享模块：

```
$ ./configure --enable-so
$ make install
```

4. 用 `apxs` 在Apache源码树以外编译并安装第三方模块，比如编译 `mod_foo.c` 为 `mod_foo.so` 的DSO模块：

```
$ cd /path/to/3rdparty
$ apxs -c mod_foo.c
$ apxs -i -a -n foo mod_foo.la
```

共享模块编译完毕后，必须在 `httpd.conf` 中用 `LoadModule` 指令使Apache启用该模块。

背景知识

现代的类Unix系统都有一种叫动态共享对象(DSO)的动态连接/加载的巧妙的机制，从而可以在运行时将编译成特殊格式的代码加载到一个可执行程序的地址空间。

加载的方法通常有两种：其一是在可执行文件启动时由系统程序 `ld.so` 自动加载；其二是在可执行程序中手动地通过Unix加载器的系统接口执行系统调用 `dlopen()/dlsym()` 进行加载。

按第一种方法，DSO通常被称为共享库(shared libraries)或者*DSO*库(DSO libraries)，使用 `libfoo.so` 或 `libfoo.so.1.2` 的文件名，存储在系统目录中(通常是 `/usr/lib`)，并在编译安装时使用连接器参数 `-lfoo` 建立了指向可执行程序的连接。通过设置连接器参数 `-R` 或者环境变量 `LD_LIBRARY_PATH` ，库中硬编码了可执行文件的路径，使Unix加载器能够在可执行程序启动时定位到位于 `/usr/lib` 目录中的 `libfoo.so` ，以解析可执行文件中尚未解析的位于DSO中的符号。

通常，DSO不会引用可执行文件中的符号(因为它是通用代码的可重用库)，也不会有后继的解析动作。可执行文件无须自己作任何动作以使用DSO中的符号，而完全由Unix加载器代办(事实上，调用 `ld.so` 的代码是被连入每个可执行文件的非静态运行时启动代码的一部分)。动态加载公共库代码的优点是明显的：只需要在系统库 `libc.so` 中存储一次库代码，从而为每个程序节省了磁盘存储空间。

按第二种方法，DSO通常被称为共享对象(shared objects)或*DSO*文件(DSO files)，可以使用任何文件名(但是规范的名称是 `foo.so`)，被存储在程序特定的目录中，也不会自动建立指向其所用的可执行文件的连接，而由可执行文件在运行时自己调用 `dlopen()` 来加载DSO到其地址空间，同时也不会进行为可执行文件解析DSO中符号的操作。Unix加载器会根据可执行程序的输出符号表和已经加载的DSO库自动解析DSO中尚未解析的符号(尤其是无所不在的 `libc.so` 中的符号)，如此DSO就获得了可执行程序的符号信息，就好象是被静态连接一样。

最后，为了利用DSO API的优点，可执行程序必须用 `dlsym()` 解析DSO中的符号，以备稍后在诸如指派表等等中使用。也就是说，可执行程序必须自己解析其所需的符号。这种机制的优点是允许不加载可选的程序部件，直到程序需要的时候才被动态地加载(也就不需要内存开销)，以扩展程序的功能。

虽然这种DSO机制看似很直接，但至少有一个难点，就是在用DSO扩展程序功能(第二种方法)时为DSO对可执行程序中符号的进行解析，这是因为，"反向解析"可执行程序中的DSO符号在所有标准平台上与库的设计都是矛盾的(库不会知道什么程序会使用它)。实际应用中，可执行文件中的全局符号通常不是重输出的，因此不能为DSO所用。所以在运行时用DSO来扩展程序功能，就必须找到强制连接器输出所有全局符号的方法。

共享库是一种典型的解决方法，因为它符合DSO机制，而且为操作系统所提供的几乎所有类型的库所使用。另一方面，使用共享对象并不是许多程序为扩展其功能所采用的方法。

截止到1998年，只有少数的软件包使用DSO机制在运行时扩展其功能，诸如 Perl 5(通过其XS机制和DynaLoader模块)，Netscape Server等。从1.3版本开始，Apache也加入此列，因为Apache已经用了基于指派表(dispatch-list-based)的方法来连接外部模块到Apache的核心。所以Apache也就当然地在运行时用DSO来加载其模块。

优点和缺点

上述基于DSO的功能有如下优点：

- 由于服务器包的装配工作可以在运行时使用 `httpd.conf` 中的配置命令 `LoadModule` 来进行，而不是在编译中使用 `编译选项` 来进行，因此显得更灵活。比如，只需要安装一个Apache，就可以运行多个不同的服务器实例(如标准&SSL版本，浓缩&功能加强版本[mod_perl、PHP])。
- 服务器可以在安装后使用第三方模块被轻易地扩展。这至少对厂商发行包的维护者有巨大的好处，他可以建立一个Apache核心包，而为诸如PHP、mod_perl、mod_fastcgi等扩展另建附加的包。
- 更简单的Apache模块原型。使用DSO配合 `apxs` ，可以脱离Apache源代码树，仅需要一个 `apxs -i` 和一个 `apachectl restart` 命令，就可以把刚开发的新模块纳入到运行中的Apache服务器。

DSO有如下缺点：

- 由于并不是所有操作系统都支持动态加载代码到一个程序的地址空间，因此DSO机制并不能用于所有平台。
- 由于Unix加载器必须进行符号解析，服务器的启动会慢20%左右。
- 在某些平台上，位置独立代码(positon independent code[PIC])需要复杂的汇编语言技巧来实现相对寻址，而绝对寻址则不需要，因此服务器在运行时会慢5%左右。
- 由于DSO模块不能在所有平台上被其他基于DSO的库所连接(`ld -lfoo`)，比如，基于a.out的平台通常不提供此功能，而基于ELF的平台则提供，因此DSO机制并不能被用于所有类型的模块。或者可以这样说，编译为DSO文件的模块只能使用由Apache核心、C库(`libc`)和Apache核心所用的所有其他动态或静态的库、含有独立位置代码的静态库(`libfoo.a`)所提供的符号。而要使用其他代码，就只能确保Apache核心本身包含对此代码的引用，或者自己用 `dlopen()` 来加载此代码。

内容协商

Apache支持HTTP/1.1规范中定义的内容协商，它可以根据浏览器提供的参数选择一个资源最合适的媒体类型、语言、字符集和编码的表现方式。它还实现了一些对浏览器发送不完整内容协商信息进行智能处理的能力。

内容协商由 `mod_negotiation` 模块支持，并被默认编译进服务器。

关于内容协商(Content Negotiation)

一个资源可能会有多种不同的表现形式，比如，可能会有不同语言或者媒体类型的版本甚至其组合。最常用的选择方法是提供一个索引页以供选择。但是由于浏览器可以在请求头信息中提供其首选项的表现形式，因此就有可能让服务器进行自动选择。比如，浏览器可以表明希望看见法语的信息，如果没有，英语的也行。如需仅请求法语的表现形式，浏览器可以发出：

```
Accept-Language: fr
```

注意：此首选项信息仅当存在多种可选的语言表现形式时才有效。

下面是一个更复杂的请求，浏览器表明，可以接受法语和英语，但最好是法语；接受各种媒体类型，最好是HTML，但纯文件或其他文本类型也可以；最好是GIF或JPEG，但其他媒体类型也可以，并允许其他媒体类型作为最终表现形式：

```
Accept-Language: fr; q=1.0, en; q=0.5
Accept: text/html; q=1.0, text/*; q=0.8, image/gif; q=0.6, image/jpeg; q=0.6, image/*;
```

Apache支持HTTP/1.1规范中定义的"服务器驱动"的内容协商， 可以完全地支持 `Accept` 、 `Accept-Language` 、 `Accept-Charset` 、 `Accept-Encoding` 请求头，这些是RFC2295和RFC2296中定义的实验协商协议，但是不支持这些RFC中定义的"功能协商"。

资源**(resource)**是一个在URI(RFC2396)中定义的概念上的实体。一个HTTP服务器，比如Apache，以表现形式**(representation)**提供对其名称空间中资源的访问，各种表现形式由已定义的媒体类型、字符集和编码的字节流构成。任何一个特定的时刻，一个资源可以没有，或者有一个，或者有多个表现形式。如果有多个表现形式存在，则称该资源是可协商的**(negotiable)**，其各种表现形式称为变种**(variant)**。而一个可协商的资源的各种变种的区别途径称为变元**(dimension)**。

Apache中的内容协商

可以使用下述两种途径之一向服务器提供有关各变种的信息，以实现对资源的协商：

- 使用类型表(也就是一个 `*.var` 文件)明确指定各变种的文件名。
- 使用"MultiViews"搜索，即服务器执行一个隐含的文件名模式匹配，并在其结果中选择。

使用类型表文件

类型表是一个与 `type-map` 处理器关联的文档(或者兼容早期Apache配置的MIME类型： `application/x-type-map`)。要使用这个功能，必须在配置中建立处理器，以定义一个文件后缀为 `type-map` ，最好的方法是在配置文件中这样设置：

```
AddHandler type-map .var
```

类型表文件应该与所描述的资源同名，且对每个有效变种都有一个块(entry)，每个块由若干连续的HTTP头行组成，不同变种的块用空行分开，块中不允许有空行。习惯上，类型表都以一个描述总体性质的组合块作为开始(这不是必须的，如果有也会被忽略)。下例是一个描述资源 `foo` 的命名为 `foo.var` 的类型表文件：

```
URI: foo

URI: foo.en.html

Content-type: text/html

Content-language: en

URI: foo.fr.de.html

Content-type: text/html;charset=iso-8859-2

Content-language: fr, de
```

注意：即使将 `MultiViews` 设置为 `On` ，类型表仍然优先于文件后缀名。如果不同的变种具有不同的资源品质，就可以对媒体类型使用"qs"参数来表示这种不同。下例演示了一个图片的jpeg, gif, ASCII-art 三个有效变种：

```
URI: foo

URI: foo.jpeg

Content-type: image/jpeg; qs=0.8

URI: foo.gif

Content-type: image/gif; qs=0.5

URI: foo.txt

Content-type: text/plain; qs=0.01
```

qs的取值范围是0.000到1.000，取值为0.000的变种永远不会被选择，没有指定qs值的变种其qs值为1.0。qs值表示一个变种相对于其他变种的"品质"，比如在表现一张照片时，jpeg通常比字符构图有更高的品质；而如果要表现的本来就是一个ASCII-art，那么当然字符构图就会比jpeg文件有更高的品质。因此，qs的值取决于变种所表现的资源本身。

mod_negotation类型表文档中有完整的HTTP头的列表。

Multiviews

`MultiViews` 是一个针对每个目录的选项，也就是说可以在 `httpd.conf` 或 `.htaccess` (如果正确设置了 `AllowOverride`)文件中的 `<Directory>` 、 `<Location>` 、 `<Files>` 配置段中，用 `Options` 指令来指定。注意， `Options All` 并不会设置 `MultiViews` ，你必须明确地指定。

`MultiViews` 的效果是：如果服务器收到对 `/some/dir/foo` 的请求，而 `/some/dir/foo` 并不存在，但是如果 `/some/dir` 启用了 `MultiViews` ，则服务器会查找这个目录下所有的foo. 文件，并有效地伪造一个说明这些*foo.* 文件的类型表，分配给他们相同的媒体类型及内容编码，并选择其中最合适的匹配返回给客户。

`MultiViews` 还可以在服务器检索一个目录时，用于 `DirectoryIndex` 指令搜索的文件名。如果设置了：

```
DirectoryIndex index
```

而 `index.html` 和 `index.html3` 并存，则服务器会作一个权衡；如果都没有，但是有 `index.cgi` ，则服务器会执行它。

如果一个目录中没有任何文件具有 `mod_mime` 可以识别的表示其字符集、内容类型、语言和编码的后缀，那么其结果将取决于 `MultiViewsMatch` 指令的设置，这个指令决定了在MultiViews协商中将使用的处理器、过滤器和其他后缀类型。

协商的方法

Apache从一个类型表或者某个目录的文件名中得到一个资源变种列表以后，会使用两种方法之一选择可能的"最佳"变种返回给客户。使用Apache的内容协商功能并不需要了解其细节，以下文档对这些方法加以详细说明，供有兴趣的人看看。

协商有两种方法：

1. 使用**Apache**算法的服务器驱动协商 是通常情况下的默认方法。使用这个算法(下面有详细的描述)，为了得到更好的效果，Apache有时会"打乱"一个特定变元(dimension)的品质因子，其方法稍后会详细阐述。
2. 透明内容协商 仅当浏览器明确地用RFC2295中定义的机制发出请求时才使用。这种方法可以给予浏览器对"最佳"变种选择的完全控制，因此其效果也取决于浏览器使用的算法。

作为透明协商过程的一部分，浏览器可以要求Apache执行RFC2296中定义的"远程变种选择算法"。

协商的变元(Dimension)

变元	说明
媒体类型	浏览器在 `Accept` 头中指明首选项，其中各项与品质因子关联，变种描述也可以有品质因子(参数"qs")。
语言	浏览器在 `Accept-Language` 头中指明首选项，其中各项与品质因子关联，变种可以与零个、一个或多个语言关联。
编码	浏览器在 `Accept-Encoding` 头中指明首选项，其中各项与品质因子关联。
字符集	浏览器在 `Accept-Charset` 头中指明首选项，其中各项与品质因子关联，变种可以指定一个字符集作为媒体类型的一个参数。

Apache协商算法

Apache使用下述算法选择可能的"最佳"变种返回给浏览器。此算法不能被再配置。其过程如下：

1. 首先，对每个协商变元，检查其适当的*Accept** 头，并对每个变种指定一个品质。如果一个变元的*Accept** 头指示不接受这个变种，则被剔除。如果最终没有变种了，则转到步骤4。
2. 顺序执行以下的测试，使用逐步剔除的方法来选择"最佳"变种。不能通过测试的变种将被剔除。每个测试完成后，如果仅剩一个变种，则作为最佳匹配，转到步骤3；如果多于一个，则继续下一个测试。
    i. 将 `Accept` 头的品质因子乘以该变种媒体类型的还原品质因子，选择乘积最高者。
    ii. 选择语言品质因子最高的变种。
    iii. 使用 `Accept-Language` 头中的语言顺序(如果存在的话)，或者使用 `LanguagePriority` 指令中的语言顺序(如果存在的话)选择最匹配的语言。
    iv. 选择最高"等级"媒体参数的变种(用以确定text/html的媒体类型)。
    v. 选择 `Accept-Charset` 头中指定的最佳字符集媒体参数的变种。如果没有明确指定，则使用ISO-8859-1字符集。具有 `text/*` 媒体类型而没有明确地与一个特定字符集关联的变种，将使用ISO-8859-1。
    vi. 选择与之关联字符集_不是_ISO-8859-1的变种，如果没有这样的变种，则选择所有的变种。
    vii. 选择最佳编码的变种。如果存在用户代理可以接受的编码的变种，则选择之；否则，如果存在混合编码的或者未编码的变种，则选择未编码的变种。如果所有的变种都是编码的，或者所有变种都是未编码的，则选择所有的变种。
    viii. 选择内容长度最小的变种。
    ix. 选择剩余变种的最前一个，这个变种或是类型表文件中的第一个，或从目录中读取

变种被时，以ASCII编码顺序的第一个文件。

3. 这时，此算法已经选择了一个"最佳"变种，并将之返回作为响应。HTTP响应头的 `Vary` 会指明协商的变元(浏览器和缓存可以利用此信息缓存该资源)。
4. 如果没有一个变种被选择(因为没有一种可以被浏览器接受)，则返回一个状态值为406响应体，并包含一个HTML格式的有效变种列表，同样，在HTTP头的 `Vary` 中指明了变种的变元。

打乱品质值

Apache有时会改变按照Apache协商算法应该被严格解析的品质值，从而在浏览器没有发送完整的精确的信息时获得更好的效果。有些很常用的浏览器在许多情况下，会发送导致变种选择错误的 `Accept` 头信息。如果一个浏览器发送了完整的且正确的信息，则不会有打乱操作。

媒体类型与通配符

`Accept:` 请求头指明了媒体类型的首选项，也可以包含"通配"媒体类型，如"image/"和匹配任何字符串的"/*"。所以，如果一个请求包含：

```
Accept: image/*, */*
```

会指明可以接受任何以"image/"开头的类型，和其他任何类型(因而前面的"image/*"就是多余的)。有些浏览器就会这样例行公事地在明确指定允许的类型后面附加通配类型，比如：

```
Accept: text/html, text/plain, image/gif, image/jpeg, */*
```

其目的是表明，明确列出的是首选项，其他不同的表现也可以。这种用法不是不可以，但是"/"其实可以通配所有其他类型，所以不推荐这样用，而应该对"."赋予一个较低的品质(首选)值0.01，如：

```
Accept: text/html, text/plain, image/gif, image/jpeg, */*; q=0.01
```

明确指定的类型没有品质值，所以其品质值是默认的最高值1.0，而"/"是较低的0.01，所以，只有在没有匹配明确指定类型的变种时，才会返回其他类型。

如果 `Accept:` 头没有指定任何q因子，那么Apache设置"/"的q值为0.01来模拟上述推荐的行为，还会设置"type/"的*q值为0.02*，使之优先于"/*"。如果 `Accept:` 头中任何媒体类型指定了q因子，则不会使用这些特殊值，以使正确发送信息的浏览器能正常运作。

语言协商的例外

在Apache 2.0中的协商算法中，新增了一些例外的规则，以允许在语言协商匹配失败的情况下，作巧妙的妥协。

通常，当客户端向服务器请求一个不能与浏览器 `Accept-language` 所匹配的唯一的页面时，服务器会返回一个"No Acceptable Variant" 或者 "Multiple Choices" 响应。但是，有可能通过配置Apache，忽略这些情况下的 `Accept-language` ，而返回一个不是非常匹配客户请求的文本，以避免这些错误信息的出现。 `ForceLanguagePriority` 指令可以屏蔽这两种错误信息，并接管由 `LanguagePriority` 指令控制的服务器裁定机制。

服务器还会在匹配失败时尝试用语言子集来匹配。例如，如果一个客户请求了一个语言是 `en-GB` 的英国英语的页面，而服务器只支持HTTP/1.1标准的简单的 `en` 。(注意，在 `Accept-Language` 中指定 `en-GB` 而不是 `en` 几乎绝对是个错误，因为它似乎暗示阅读的人懂英国英语却不懂大众英语。而不幸的是，许多流行的客户端的默认配置却是这样的)。如果没有可以匹配的语言，服务器将会忽略其语言子集的设定，返回"No Acceptable Variants"错误，或者按 `LanguagePriority` 指令作妥协。Apache会隐含地在客户可接受语言的列表中附加一个具有很低品质值的父语言，但是，如果客户请求"en-GB; q=0.9, fr; q=0.8" 那么将返回"fr"的文本，这对遵循HTTP/1.1标准以使正确配置的浏览器能正常工作是必须的。

为了支持用于确定用户首选语言的高级技术(比如cookies或特殊的URL路径)，从2.0.47版本起 `mod_negotiation` 模块开始支持 `prefer-language` 环境变量`模块将会尝试选择一个匹配的变种。如果不存在这样的变种，将会使用上述通常的协商过程。

示例

```
SetEnvIf Cookie "language=(.+)" prefer-language=$1
```

透明内容协商的扩展

Apache在变种列表中使用了一个新的 `{encoding ..}` 元素来标记变种，从而扩展了透明内容协商协议(RFC2295)。实现RVSA/1.0算法(RFC2296)的目的是识别列表中被编码的变种，作为可以被 `Accept-Encoding` 请求头接受的候选变种。在选择最佳变种之前，RVSA/1.0的实现不会对品质因子作四舍五入的运算。

超链和名称转换说明

如果使用语言协商，由于文件可以有不止一个后缀，因此就可以选择不同的名称转换，其后缀顺序通常是无关紧要的(参见mod_mime文档)。

一个典型的有MIME类型后缀的文件(如 `html`)，其后缀可以是编码后缀(如 `gz`)，也可以是语言变种后缀(如 `en`)

例如：

- foo.en.html
- foo.html.en
- foo.en.html.gz

文件名和有效及无效超链的例子：

文件名	有效超链	无效超链
foo.html.en	foo foo.html	-
foo.en.html	foo	foo.html
foo.html.en.gz	foo foo.html	foo.gz foo.html.gz
foo.en.html.gz	foo	foo.html foo.html.gz foo.gz
foo.gz.html.en	foo foo.gz foo.gz.html	foo.html
foo.html.gz.en	foo foo.html foo.html.gz	foo.gz

可以看出，上表中使用没有任何后缀的超链(如 `foo`)总是可行的，其优点是可以隐藏rsp. 文件的真实类型，而可以在将来作更改，比如，不用修改超链本身，而改变 `html` 为 `shtml` 或 `cgi` 。

如果希望在超链中继续使用MIME类型(如 `foo.html`)，则语言后缀(还包括一个编码后缀)必须出现在MIME类型后缀的右边(如 `foo.html.en`)。

缓冲说明

如果缓存中有一个与特定URL关联的表现形式(representation)，那么下一次该URL被请求时，缓存就可以使用它。但是，如果这个资源在服务器端是可协商的，则可能只有第一次请求的变种是正确的，而其后由于缓存中命中而取出的结果是错误的。为避免这种情况的发生，Apache通常把内容协商之后返回的响应标记为不可以被HTTP/1.1客户端缓冲。另外Apache还支持HTTP/1.1协议的功能以允许缓冲已协商的请求。

对来自HTTP/1.0客户端的请求(浏览器或缓存)， `CacheNegotiatedDocs` 指令可以允许缓存服从协商的请求。此指令应该出现在主服务器或虚拟主机的配置中，没有参数，并且对来自HTTP/1.1客户端的请求没有影响。

对于遵守HTTP/1.1规范的客户端，Apache发送一个 `Vary` 应答头以指定该应答的协商变元。缓存可以使用这个信息来判断一个其后的请求是否可以从本地副本中提供服务。为了鼓励缓存使用本地副本而不是协商变元，请设置 `force-no-vary` 环境变量。

更多信息

更多有关内容协商的信息，可以参见Alan J. Flavell的Language Negotiation Notes，但是注意，此文档可能没有升级以包含Apache2.0中的改变。

自定义错误响应

Apache可以让网站管理员自己自定义对一些错误和问题的响应。

自定义的响应可以定义为当服务器检测到错误或问题时才被激活。

如果一个脚本崩溃并产生"500 Server Error"响应，那么这个响应可以被更友好的提示替换或者干脆用重定向语句跳到其他的URL(本地的或外部的)。

行为

老式的行为

Apache1.3 会响应一些对于用户没有任何意义的错误或问题信息，而且不会将产生这些错误的原因写入日志。

新式的行为

服务器可以被要求作出如下应答：

1. 显示一些其他的文字以代替硬编码的信息
2. 重定向到本地URL
3. 重定向到一个外部的URL

当一些信息可以被传递的时候，重定向到另外一个URL就变得很有用。这些信息用于更清楚的解释和/或记录一些错误或问题产生的原因。

为了达到这个目的，Apache将定义一些新的类似于CGI环境变量的环境变量：

```
REDIRECT_HTTP_ACCEPT=*/*, image/gif, image/x-xbitmap, image/jpeg

REDIRECT_HTTP_USER_AGENT=Mozilla/1.1b2 (X11; I; HP-UX A.09.05 9000/712)

REDIRECT_PATH=.:/bin:/usr/local/bin:/etc

REDIRECT_QUERY_STRING=

REDIRECT_REMOTE_ADDR=121.345.78.123

REDIRECT_REMOTE_HOST=ooh.ahhh.com

REDIRECT_SERVER_NAME=crash.bang.edu

REDIRECT_SERVER_PORT=80

REDIRECT_SERVER_SOFTWARE=Apache/0.8.15

REDIRECT_URL=/cgi-bin/buggy.pl
```

请注意"`REDIRECT_`"这个前缀。

至少会有`REDIRECT_URL`和`REDIRECT_QUERY_STRING`两个变量会被传递到新的URL(假定这个URL是cgi脚本或者是cgi包含页面)。其他变量将仅在发生错误或问题之前存在的情况下才存在。如果你的`ErrorDocument`使用了外部重定向(任何类似于`http:`开头的形式，哪怕它仍指向同一个服务器)，将没有任何变量被指定。

配置

当对`ErrorDocument`进行了相应的设置后，将可以在`.htaccess`文件中使用`AllowOverride`指令。

以下是一些示例...

```
ErrorDocument 500 /cgi-bin/crash-recover

ErrorDocument 500 "Sorry, our script crashed. Oh dear"

ErrorDocument 500 http://xxx/

ErrorDocument 404 /Lame_excuses/not_found.html

ErrorDocument 401 /Subscription/how_to_subscribe.html
```

语法如下：

`ErrorDocument <3位错误代码> <action>`

<action>可以代表：

1. 用于显示的用双引号(")界定的文字。双引号之间的所有文字都将被显示，但双引号本身不会被显示
2. 作为重定向目的外部URL
3. 作为重定向目的本地URL

自定义错误响应与重定向

Apache重定向到URL的行为已经进行了修改，以便可以在脚本/服务器端包含页面加入额外的环境变量。

老式的行为

标准CGI变量对于重定向的目的脚本来说是可见的。但没有说明重定向的来源。

新式的行为

一批新的环境变量将被初始化并提供给重定向的目标脚本。每个新变量都有一个" `REDIRECT_` "前缀。 `REDIRECT_*` 环境变量由重定向之前的CGI环境变量创建而来，并被加上了" `REDIRECT_` "前缀。比如说， `HTTP_USER_AGENT` 变成了 `REDIRECT_HTTP_USER_AGENT` 。在这些新变量之外，Apache还将定义 `REDIRECT_URL` 和 `REDIRECT_STATUS` 来帮助脚本确定重定向的来源。重定向的源URL和目的URL都能被记录到访问日志中。

如果 `ErrorDocument` 指定了一个到本地CGI脚本的重定向，该脚本应当在它的输出中包含一个" `Status:` "头字段以确保将导致调用它的错误条件始终返回客户端。举例来说，一个Perl `ErrorDocument` 脚本可能包含如下内容：

```
...
print  "Content-type: text/html\n";
printf "Status: %s <中断条件>\n", $ENV{"REDIRECT_STATUS"};
...
```

如果该脚本专门用于处理一个特定的错误条件，比如： `404 Not Found` ，它就可以使用特定的代码和错误文本进行替代。

需要注意的是如果应答包含一个" `Location:` "头(为了进行一个客户端重定向)，脚本必须发出一个适当的" `Status:` "头(比如： `302 Found`)。否则" `Location:` "头可能无效。

地址和端口的绑定(Binding)

配置Apache监听指定的地址和端口。

概述

相关模块

- `core`
- `mpm_common`

相关指令

- `<VirtualHost>`
- `Listen`

Apache启动时，会绑定本机上的地址和端口，然后等待请求的进入。默认情况下，它会监听本机的所有地址。但是，当需要监听特定的地址或端口或地址与端口的组合，或者需要对不同的IP地址、主机名、端口作出不同的响应(如使用虚拟主机)时，就必须明确指定。

`Listen` 指令告诉服务器接只受来自特定端口(或地址+端口的组合)的请求。如果 `Listen` 指令仅指定了端口，则服务器会监听所有的IP地址；如果指定了地址+端口的组合，则服务器只监听来自此特定地址上特定端口的请求。使用多个 `Listen` 指令，可以指定在多个地址和端口上进行监听。

例如：使服务器同时接受来自端口80和8000的请求，可以这样写：

```
Listen 80
Listen 8000
```

接受来自两个指定的地址+端口的组合：

```
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

IPv6地址必须用方括号括起来：

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

针对IPv6的特殊考虑

有越来越多的平台开始支持IPv6，而APR在大多数平台上也支持IPv6，使Apache能够获得IPv6套接字，并处理通过IPv6发送的请求。

一个经常令Apache管理员疑惑的问题是IPv6的套接字能否同时处理IPv4和IPv6的连接。IPv6套接字在处理IPv4连接时使用的是将IPv6映射到IPv4的地址(IPv4-mapped IPv6 addresses)，这样做在大多数平台上默认是允许的，而在FreeBSD、NetBSD、OpenBSD上，为了配合其系统全局策略，默认却是禁止的。即使在这些默认禁止的平台上，Apache也可以通过特殊的 `编译选项` 来改变这种行为。

另一方面，在某些平台上(如Linux和Tru64)同时处理IPv6和IPv4的唯一方法就是使用映射地址(mapped addresses)。如果你希望Apache以最少的套接字同时处理IPv4和IPv6的连接，就必须使用映射到IPv4的IPv6地址，也就是必须指定 `--enable-v4-mapped` `编译选项` 。

在除FreeBSD、NetBSD、OpenBSD以外的其他平台上， `--enable-v4-mapped` 是编译时的默认值。因此你正在使用的Apache很可能就是按照这种方式编译的。

要使Apache仅仅只处理IPv4连接，无论你的平台是什么或者APR是否支持，只须对所有 `Listen` 指令都指定IPv4地址即可，如下所示：

```
Listen 0.0.0.0:80

Listen 192.170.2.1:80
```

如果你的平台支持从IPv6到IPv4的地址映射，但是你又希望Apache使用不同的套接字分别处理IPv4和IPv6的连接(也就是禁用地址映射)，必须明确指定 `--disable-v4-mapped` `编译选项` 。注意： `--disable-v4-mapped` 在FreeBSD、NetBSD、OpenBSD上是默认值。

怎样与虚拟主机协同工作

`Listen` 指令并不实现虚拟主机，它只是告诉主服务器(main server)去监听哪些地址和端口。如果没有 `<VirtualHost>` 指令，服务器将对所有请求一视同仁；但是如果有 `<VirtualHost>` 指令，则服务器会对不同的地址和端口作出不同的响应。要实现虚拟主机，首先必须告诉服务器需要监听哪些地址和端口，然后为每个特定的地址和端口建立一个 `<VirtualHost>` 段来执行特定的相应。注意，如果将 `<VirtualHost>` 段设置为服务器没有监听的地址和端口，则此段无效。

多路处理模块

本文档描述了什么是多路处理模块以及它如何为Apache所使用。

简介

Apache HTTP服务器被设计为一个强大的、灵活的能够在多种平台以及不同环境下工作的服务器。不同的平台和不同的环境经常产生不同的需求，或是为了达到同样的最佳效果而采用不同的方法。Apache凭借它的模块化设计很好的适应了大量不同的环境。这一设计使得网站管理员能够在编译时和运行时凭借载入不同的模块来决定服务器的不同附加功能。

Apache2.0将这种模块化的设计延伸到了web服务器的基础功能上。这个版本带有多路处理模块(MPM)的选择以处理网络端口绑定、接受请求并指派子进程来处理这些请求。

将模块化设计延伸到这一层次主要有以下两大好处：

- Apache可以更简洁、更有效地支持各种操作系统。尤其是在 `mpm_winnt` 中使用本地网络特性代替Apache1.3中使用的POSIX模拟层后，Windows版本的Apache现在具有更好的性能。这个优势借助特定的MPM同样延伸到了其他各种操作系统。
- 服务器可以为某些特定的站点进行定制。比如，需要更好伸缩性的站点可以选择象 `worker` 或 `event` 这样线程化的MPM，而需要更好的稳定性和兼容性以适应一些旧的软件的站点可以用 `prefork` 。

从用户角度来看，MPM更像其他的Apache模块。主要的不同在于：不论何时，必须有且仅有一个MPM被载入到服务器中。现有的MPM列表可以在模块索引中找到。

选择一个MPM

MPM必须在编译配置时进行选择，并静态编译到服务器中。如果编译器能够确定线程功能被启用，它将会负责优化大量功能。因为一些MPM在Unix上使用了线程，而另外一些没有使用，所以如果在编译配置时选择MPM并静态编译进Apache，Apache将会有更好的表现。

你可以在使用 `configure` 脚本时用 `--with-mpm=_NAME_` 选项指定MPM，*NAME*就是你想使用的MPM的名称。

一旦服务器编译完成，就可以用 `./httpd -l` 命令来查看使用了哪个MPM。这个命令将列出所有已经被编译到服务器中的模块，包括MPM。

默认的MPM

下表列出了不同操作系统上默认的MPM。如果你在编译时没有进行选择，这将是默认选择的MPM。

BeOS	`beos`
Netware	`mpm_netware`
OS/2	`mpmt_os2`
Unix	`prefork`
Windows	`mpm_winnt`

Apache的环境变量

Apache HTTP服务器提供了一个机制，可以把信息存储在叫做环境变量(environment variable)的命名变量中。这个信息可以用于控制诸如日志记录和访问控制之类的操作。此外，还可以作为一个和诸如CGI脚本这样的外部程序进行沟通的机制。本文档讨论了操作和使用这些变量的不同方法。

尽管这些变量也被称作环境变量，但它们和底层的、由操作系统控制的环境变量不能混为一谈。这些变量仅在Apache内部被存储和操纵。仅当它们被提供给外部CGI脚本或服务器端包含脚本(SSI)时，才会变成真正的操作系统环境变量。如果你想操作作为服务器运行基础的操作系统的环境变量，你必须使用由你的操作系统shell提供的标准环境操作机制。

设置环境变量

相关模块

- `mod_env`
- `mod_rewrite`
- `mod_setenvif`
- `mod_unique_id`

相关指令

- `BrowserMatch`
- `BrowserMatchNoCase`
- `PassEnv`
- `RewriteRule`
- `SetEnv`
- `SetEnvIf`
- `SetEnvIfNoCase`
- `UnsetEnv`

基本的环境变量操作

设置一个Apache环境变量最基本的方法，就是使用没有什么限制的 `SetEnv` 指令。也可以使用 `PassEnv` 指令将启动Apache的操作系统shell的环境变量传进来。

针对每个请求进行有条件的设定

为了具有额外的伸缩性， `mod_setenvif` 提供的指令允许针对每个请求特定的请求特性进行环境变量的设定。比如，可以仅在一个特定的浏览器(User-Agent)进行请求时，或仅在一个特定的"Referer:"头被发现时进行环境变量的设置。如果使用 `mod_rewrite` 的 `RewriteRule` 指令中的 `[E=...]` 选项来进行环境变量的设置，还会具有更大伸缩性。

唯一标识符

最后， `mod_unique_id` 将为每个请求设定一个 `UNIQUE_ID` 环境变量的值，这个值对"所有"请求都是唯一的，即使在极为特定的条件下。

标准CGI变量

除了所有Apache配置中的环境变量和由操作系统shell传进来的环境变量之外，还有一组环境变量是提供给CGI脚本和SSI页面的，此组环境变量包含由CGI规范要求的与请求相关的元信息。

一些告诫

- 用环境变量操作指令来覆盖或修改标准的CGI变量是行不通的。
- 当用 `suexec` 来运行CGI脚本时，环境变量将会被清除到在CGI脚本运行之前只剩一组安全变量。安全变量的列表在编译时由suexec.c定义。
- 出于可移植性的考虑，环境变量的命名必须仅包含字母、数字、下划线。此外，第一个符号不能为数字。不符合此要求的字符将在传递给CGI脚本和SSI页面时被下划线取代。

使用环境变量

相关模块

- `mod_authz_host`
- `mod_cgi`
- `mod_ext_filter`
- `mod_headers`
- `mod_include`
- `mod_log_config`
- `mod_rewrite`

相关指令

- `Allow`
- `CustomLog`
- `Deny`

- `ExtFilterDefine`
- `Header`
- `LogFormat`
- `RewriteCond`
- `RewriteRule`

CGI脚本

环境变量的主要用途之一就是把信息传递给CGI脚本。如前所述，递给CGI脚本的环境变量，除了在Apache配置中定义的以外，还包含一组与请求相关的标准元信息的环境变量。更多细节请参见CGI教程。

SSI页面

由mod_include的 `INCLUDES` 过滤器处理的服务器端解析(Server-parsed[SSI])文档能够用 `echo` 元素打印出环境变量，并能在流程控制元素中使用环境变量来基于请求特性而产生部分页面。Apache当然也会将上述的标准CGI环境变量提供给SSI页面。更多细节请参见SSI教程。

访问控制

可以用 `allow from env=` 和 `deny from env=` 指令基于环境变量的值对服务器进行访问控制。在结合了 `SetEnvIf` 之后，能更灵活的基于客户端特性对服务器进行访问控制。比如，你能用这些指令来拒绝一些特定浏览器(User-Agent)的访问。

条件日志记录

可以用 `LogFormat` 的可选项" `%e` "将环境变量写入访问日志中。此外，还可以用 `CustomLog` 指令基于环境变量的状态来决定是否将请求写入日志。在结合了 `SetEnvIf` 之后，能更灵活的控制哪些请求将被记录。比如，你可以选择不对以 `gif` 为结尾的文件名请求进行记录，或者选择只记录内网之外的客户端请求。

条件响应头

`Header` 指令能根据一个环境变量是否存在来决定是否将一个HTTP头放入对客户端的响应里。这将使诸如从客户端收到特定的请求头时返回特定的应答头这样的事情成为可能。

外部过滤器的激活

由 `mod_ext_filter` 的 `ExtFilterDefine` 指令配置的外部过滤器可以用 `disableenv=` 和 `enableenv=` 选项根据环境变量的条件进行激活。

URL重写

`RewriteCond` 中形如 `%{ENV:...}` 的*TestString*允许mod_rewrite的重写引擎以环境变量为条件进行决策。注意：mod_rewrite内部可以访问但没有以 `ENV:` 开头的那些变量并不是真正的环境变量。它们只是mod_rewrite特有的变量而不能被其他模块所访问。

用于特殊目的的环境变量

由于互操作性的问题，在针对特定客户端的处理中，引入了一套修正Apache行为的机制。为了使这些机制尽量灵活，它们将通过环境变量的定义而激活。比如，典型的示例有 `BrowserMatch` ，尽管 `SetEnv` 和 `PassEnv` 也能使用。

downgrade-1.0

即使这个请求符合更新的标准，也强制把它当作一个HTTP/1.0请求来处理。

force-gzip

如果你激活了 `DEFLATE` 过滤器，这个环境变量将忽略浏览器的accept-encoding设置而无条件的使用经过gzip压缩的输出。

force-no-vary

此变量在将应答送回客户端之前删除所有的 `Vary` 头字段。一些客户端不能正确地解析此头字段。此变量的设定将解决此问题，它同时隐含设置了**force-response-1.0**。

force-response-1.0

设定该变量可以在客户端发送HTTP/1.0请求时，强制进行HTTP/1.0响应。它的实现源于一个AOL的代理产生的问题。一些HTTP/1.0客户端在收到HTTP/1.1的响应后会有不正常的举动。而设定此变量能够解决这一问题。

gzip-only-text/html

当该变量为"1"时，将禁止 `text/html` 之外的内容类型使用由 `mod_deflate` 提供的 `DEFLATE` 输出过滤器。如果你更喜欢使用静态的压缩文件； `mod_negotiation` 也同样使用该变量(不单单是gzip，而是所有不具有"同一性"的编码)。

no-gzip

如果设置了此变量，`mod_deflate` 中的 `DEFLATE` 过滤器将被禁用，同时 `mod_negotiation` 将拒绝发送经过编码的资源。

nokeepalive

如果设置了此变量，`KeepAlive` 将被禁用。

prefer-language

此变量将影响 `mod_negotiation` 的行为。如果它包含一个语言标签(如：`en`、`fr`、`zh_cn`、`x-方言`)，`mod_negotiation` 将尝试发送一个标签指定的语言的变种，如果不存在这样的变种，则使用通常的内容协商处理过程。

redirect-carefully

此变量将使服务器在对客户端发送重定向命令时更加小心。典型应用于已知客户端在处理重定向指令时会存在问题的情况下。它的实现源于微软的WebFolders软件存在的一个问题。它在经由DAV方法在目录资源上处理重定向命令时会有问题。

suppress-error-charset

仅存在于2.0.54后的版本中

当Apache针对用户请求响应一个重定向命令的时候，这个响应中包含了一些文字。这些文字将在客户端不能(或没有)自动执行重定向操作的情况下显示。Apache会将这段文字按照ISO-8859-1字符集进行编码。

然而，如果重定向的目的页面使用了不同的字符集，一些有问题的浏览器版本会使用重定向命令文本的字符集，而不是采用目的页面的字符集。比如，希腊文就不会被正确显示。

设置此环境变量将使Apache略过重定向命令文本的字符集设置，这样这些有问题的浏览器就会正确的使用目的页的字符集。

force-proxy-request-1.0, proxy-nokeepalive, proxy-sendchunked, proxy-sendcl

这些指令改变了 `mod_proxy` 协议的行为，参见 `mod_proxy` 文档以获得更多细节。

示例

针对表现不恰当的客户端改变协议的行为

早期的版本建议将以下示例包含到httpd.conf中以解决一些已知的客户端问题。但是这些存在问题的客户端现在基本上已经绝种了，所以，下列示例也就没有存在的必要了。

```
# 下面的指令将会修改HTTP的普通响应方式。
# 第一个指令为Netscape 2.x浏览器禁用keepalive特性，因为它不能正确处理。
# 第二个指令用于IE4.0，因为它也不能对HTTP/1.1的301/302(重定向)应答正确处理keepalive。
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0

# 下面的指令为违反HTTP/1.0规范的浏览器禁用HTTP/1.1应答。
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
```

不在访问日志中记录对图片的请求

以下示例将避免将对图片的请求记入访问日志中。你修改一下就可以将它用于避免特定目录或特定主机的请求被记入日志。

```
SetEnvIf Request_URI \.gif image-request
SetEnvIf Request_URI \.jpg image-request
SetEnvIf Request_URI \.png image-request
CustomLog logs/access_log common env=!image-request
```

阻止"图片大盗"

下例展示了如何避免不在你服务器上的人在他们的站点中直接引用你服务器上的图片。这不是一个推荐的配置，但它能在有限的环境中加以应用。我们假设你所有的图片都在/web/images目录下。

```
SetEnvIf Referer "^http://www.example.com/" local_referal
# 允许未发送Referer头的浏览器
SetEnvIf Referer "^$" local_referal
<Directory /web/images>
   Order Deny,Allow
   Deny from all
   Allow from env=local_referal
</Directory>
```

想得知此技术的更多信息，请参阅"今日Apache教程"《保护你的图片不为他人所用》。

Apache处理器的使用

本文阐述Apache处理器的使用。

什么是处理器(Handler)

相关模块

- `mod_actions`
- `mod_asis`
- `mod_cgi`
- `mod_imagemap`
- `mod_info`
- `mod_mime`
- `mod_negotiation`
- `mod_status`

相关指令

- `Action`
- `AddHandler`
- `RemoveHandler`
- `SetHandler`

"处理器"是当一个文件被调用时，Apache所执行操作的内部表现。文件一般都有基于其文件类型的隐含处理器。通常，文件都只是被服务器简单的提交，只有某些文件类型会被特别地"处理"。

Apache1.1增加了使用处理器的能力。处理器可以基于文件名后缀或位置进行指定，而不只是文件类型，其优越性不仅在于它是一个优秀的方案，还在于它允许一个文件同时与一种类型和一个处理器相关联。(参见：带多扩展名的文件)

处理器可以被编译进服务器也可以包含在模块中，还可以用 `Action` 指令增加。标准发行版中内建的处理器如下：

- **default-handler**：使用 `default_handler()` 发送文件，这是处理静态内容的默认处理器。(`core`)
- **send-as-is**：按原样带HTTP头发送文件。(`mod_asis`)
- **cgi-script**：将文件视为CGI脚本。(`mod_cgi`)
- **imap-file**：将文件作为映射表规则文件解析。(`mod_imagemap`)
- **server-info**：获取服务器的配置信息。(`mod_info`)

- **server-status**：获取服务器状态的报告。(`mod_status`)
- **type-map**：将文件作为类型表文件解析以实现内容协商。(`mod_negotiation`)

例子

用CGI脚本修改静态的内容

以下指令，将使对带有 `html` 后缀的文件的请求，调用CGI脚本 `footer.pl`

```
Action add-footer /cgi-bin/footer.pl

AddHandler add-footer .html
```

然后，由CGI脚本负责发送(由环境变量 `PATH_TRANSLATED` 指向的)原始请求文档，并按需要进行修改或增加。

带HTTP头的文件

以下指令启用 `send-as-is` 处理器，它用于处理本身包含HTTP头的文件，这样，所有位于 `/web/htdocs/asis/` 目录中的文件，无论其后缀名是什么，都由 `send-as-is` 进行处理。

```
<Directory /web/htdocs/asis>

SetHandler send-as-is

</Directory>
```

程序员注意事项

为了实现处理器功能，Apache API里面增加了一些内容，你可能会用到。尤其是Apache API结构中增加了一个字段：

```
char *handler
```

如果你的模块需要使用处理器，只须在对请求执行 `invoke_handler` 之前，设置 `r->handler` 为该处理器的名称即可。处理器的实现和以前一样，只是使用了处理器名称而不是内容类型。处理器的名称可以有"-"，但不能有"/"，以避免和介质类型名称冲突。

过滤器(Filter)

本文阐述Apache中过滤器的用法。

Apache 2 中的过滤器

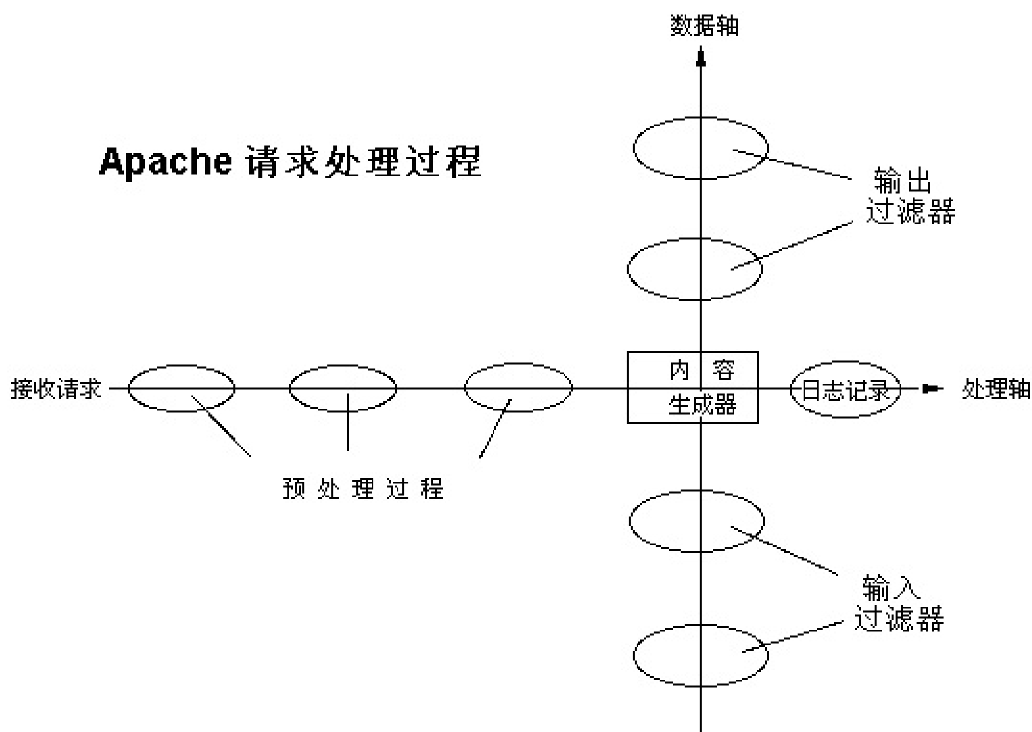
相关模块

- `mod_filter`
- `mod_deflate`
- `mod_ext_filter`
- `mod_include`
- `mod_charset_lite`

相关指令

- `FilterChain`
- `FilterDeclare`
- `FilterProtocol`
- `FilterProvider`
- `AddInputFilter`
- `AddOutputFilter`
- `RemoveInputFilter`
- `RemoveOutputFilter`
- `ExtFilterDefine`
- `ExtFilterOptions`
- `SetInputFilter`
- `SetOutputFilter`

Apache 2.0 及以后的版本中使用了过滤器链，使得应用程序能够以高度灵活的、可配置的方式处理进入的数据和输出的数据，而无需关心这些数据来自哪里。我们可以预处理进入的数据和后处理(post-process)输出的数据。这些过程基本上独立于传统的请求处理阶段。



标准Apache发行版中的一些过滤器实例：

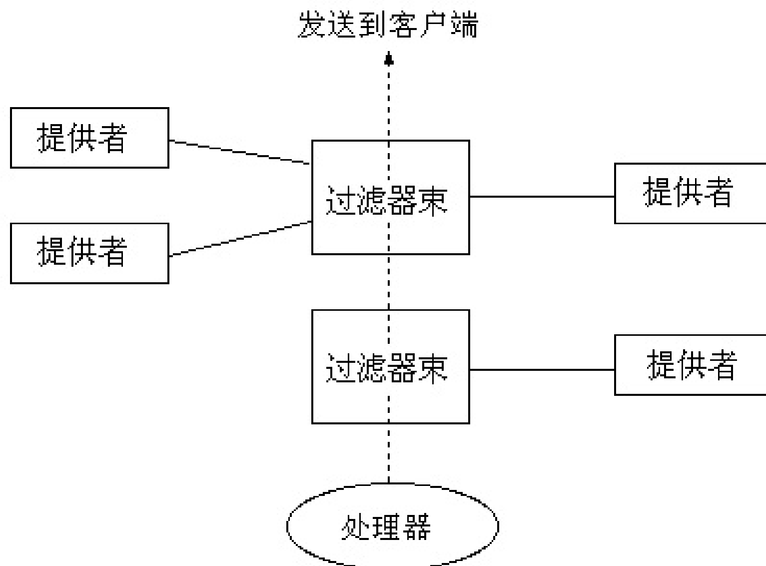
- `mod_include` 实现了服务端包含。
- `mod_ssl` 实现了SSL加密(https)。
- `mod_deflate` 实现了实时压缩/解压。
- `mod_charset_lite` 在不同的字符集之间自动转换。
- `mod_ext_filter` 将一个外部程序作为过滤器运行。

Apache还在内部使用许多过滤器来对请求执行诸如分块、字节层次处理(byte-range handling)等功能。

可以从modules.apache.org或别处得到许多通过第三方过滤器模块实现的应用程序。其中的一些是：

- HTML和XML处理和重写
- XSLT变换XIncludes
- XML名字空间支持
- 文件上传处理和HTML表格解码
- 图像处理
- 保护诸如PHP脚本之类的脆弱应用程序
- 文本搜索和替换编辑

智能过滤



Apache 2.1 及以后的版本中包含的 `mod_filter` 模块允许过滤器链在运行时进行动态配置。举例来说，你可以在代理无法得到原始服务器发送的初始信息的情况下，安装一个使用不同的过滤器来分别重写HTML、处理JPEG图片的代理。这是通过使用一个过滤器束(filter harness)来工作的，该过滤器束在运行时根据实际内容的不同将任务分配给不同的提供者(provider)。任何一个过滤器都何以被直接插入到过滤器链中并无条件的运行，或者动态的作为一个提供者(provider)插入。例如：

- HTML处理过滤器仅在内容是text/html或application/xhtml+xml的情况下才运行
- 压缩过滤器仅在输入是一个未经压缩的可压缩类型的内容的情况下才运行
- 字符集转换过滤器仅在一个文本文档不是期望的字符集的情况下才被插入

使用过滤器

有两种使用过滤器的方法：简单方法和动态方法。通常，你应当使用两者之一；混合使用它们可能会导致意想不到的后果(虽然简单方法的输入过滤器可以和简单或动态方法的输出过滤器混合使用)。

简单方法是配置输出过滤器的唯一方法，并且对于需要静态过滤器链的输出过滤器来说是足够的。相关的指令有： `SetInputFilter` ， `SetOutputFilter` ， `AddInputFilter` ， `AddOutputFilter` ， `RemoveInputFilter` ， `RemoveOutputFilter` 。

动态方法使输出过滤器能够进行静态的或者灵活的动态配置成为可能，详情请参见 `mod_filter` 模块文档。相关指令有： `FilterChain` ， `FilterDeclare` ， `FilterProvider` 。

一个更高级的指令 `AddOutputFilterByType` 也被支持，但是可能存在一些问题，目前还不赞成使用它，而代之以动态配置。

suEXEC支持

suEXEC特性使得Apache可以使用与调用web服务器的用户不同的用户身份来运行**CGI**和**SSI**程序。而通常情况下，CGI或者SSI程序执行时使用和web服务器相同的用户身份。

正确运用该特性，可以减少很多因为提供用户执行私有CGI或者SSI程序所带来的安全风险。但如果配置不当的话，则可能引起很多问题，使你的计算机产生更多的安全漏洞。如果你对管理 *setuid root* 程序以及可能导致的安全问题不熟悉的话，我们强烈建议你不要使用suEXEC。

开始之前

在我们开始切入正题之前，你必须明白Apache开发组以及本文档所做的假设。

首先，我们假设你正在使用类UNIX操作系统，只有这类操作系统才具有**setuid**和**setgid**命令。所有的其他命令行的例子也是如此。其他的操作系统平台，即使也支持suEXEC，但是它的配置可能和我们所讲的并不相同。

第二，我们假设你熟悉计算机的安全和管理计算机的一些基本概念。这关系到如何正确理解**setuid/setgid**操作以及对你的系统可能带来的各种影响和不同的安全等级。

第三，我们假设你正在使用源代码未经修改的suEXEC版本。所有suEXEC的代码都经过开发者的仔细查验并做过大量测试。在这些代码中，人们采取了各种预防措施，使之简单、健壮、安全。修改这些代码可能会导致预料之外的问题和安全隐患。所以我们强烈地建议你不要修改代码，除非你精通安全编程，并愿意和Apache开发组共享成果。

第四，也是最后一点，Apache开发组已经决定默认不安装suEXEC。suEXEC的配置需要管理员细致关注各个细节。在仔细考察过关于suEXEC的各种设置方法后，管理员应该使用标准的安装方法来安装suEXEC。设置的参数应该经过仔细推敲，以保证系统的安全运行。Apache开发组希望通过限制suEXEC的安装，仅使那些经过细致理解，并有能力运用它的管理员来使用。

你还想使用suEXEC吗？还想？很好！那我们开始吧！

suEXEC的安全模型

在我们开始配置和安装suEXEC之前，我们需要先讨论一下它的安全模型。这样，你才能更好的理解suEXEC内部究竟做了些什么事情，以及哪些确保系统安全的预防措施。

suEXEC是基于一个setuid的"封装"程序，该程序由"主"Apache web服务器调用。当一个HTTP请求的是管理员指定的、以不同于"主"服务器用户身份运行的CGI或SSI程序时，该封装程序将被调用。处理这样的请求时，Apache将被请求的程序名及其UID和GID提供给suEXEC封装器。

封装器(wrapper)通过处理下面所描述的步骤，来决定封装的成功或失败：如果有任意一个条件为假，程序将把错误情况记录到日志中，退出并返回错误信息。否则继续执行。(以下所说的"程序"均指"CGI/SSI程序")

1. 用户使用了合法的系统账号来执行封装程序了吗？

   确保运行封装器的是一个系统中确实存在的用户。

2. 封装器被调用时，使用的参数个数正确吗？

   封装器仅在使用了正确数量的参数调用时才会执行。Apache web服务器知道正确的参数格式是什么。如果封装器没有收到正确数量的参数，则说明要么被黑客攻击，要么Apache二进制代码中suEXEC的部分出了问题。

1. 这个合法的用户被允许运行封装器吗？

   这个用户是可以运行封装器的用户吗？仅有一个用户(Apache用户)被允许运行封装器。

2. 目标**CGI/SSI**程序有不安全的分级路径索引吗？

   目标CGI/SSI程序包含了"/"开头或者有".."后向路径索引吗？这些都是不允许的；并且目标程序必须位于suEXEC的文档根目录下。(参见下面的：`--with-suexec-docroot=_DIR_`)

3. 目标程序的所属用户名合法吗？

   目标程序的所属用户名存在吗？

4. 目标程序的所属用户组合法吗？

   目标程序的所属用户组存在吗？

5. 目标程序的所属用户名不是超级用户吗？

   目前，suEXEC不允许root执行CGI/SSI程序。

6. 目标程序所属用户的**UID**高于最小**UID**值吗？

   最小UID值是在配置中指定的。你可以指定允许执行CGI/SSI程序的最小UID值，这样可以保证不会和系统账号冲突。

7. 目标程序的所属用户组不是超级用户组吗？

   目前，suEXEC不允许root组用户执行CGI/SSI程序。

8. 目标程序所属用户组的**GID**高于最小**GID**值吗？

最小GID值是在配置中指定的。你可以指定允许执行CGI/SSI程序的最小GID值，这样可以保证不会和系统账号冲突。

9. 封装器能够成功地变为目标用户和组吗？

   这里就是程序变为目标用户和组的关键步骤了。我们是通过调用setuid和setgid来实现的。在组访问列表中，和该用户相关的所有组信息都将被初始化。

10. 能够切换到程序所在的目录吗？

    如果不存在，将无法包含程序文件。如果不能切换一般也表示目录不存在。

11. 这个目录在**Apache**的网络空间中吗？

    如果是对于服务器的一般请求，那么请求的目录是在suEXEC的根文档目录下吗？如果请求的是一个用户目录，那么该目录是在suEXEC配置的该用户的根目录下吗？(参见：suEXEC配置选项)

12. 该目录不具有其他用户可写的权限吗？

    我们不想把目录开放给其他人；只有属主才可以改变该目录中的内容。

13. 目标**CGI/SSI**程序存在吗？

    如果不存在，当然无法继续运行。

14. 目标**CGI/SSI**程序不可以被其他用户改写吗？

    我们不想给其他人有修改程序的权限。

15. 目标程序尚未被**setuid**或者**setgid** ？

    我们不想要执行的程序被再次改变UID/GID。

16. 目标用户和组与程序的用户和组相同吗？

    用户是这个文件的属主吗？

17. 我们可以成功清除进程的环境变量并保证操作的安全性吗？

    suEXEC通过建立一个安全的可执行路径(在配置中定义)来清除该进程的环境变量，同时只传送在安全环境变量列表(配置中定义)中所列出的环境变量。

18. 可以成功的变为目标程序并执行吗？

    这里就是suEXEC结束，并开始运行目标程序的地方了。

这是suEXEC封装器标准操作方式的安全模型。它有些严格，并强加了CGI/SSI设计上的限制。但它是仔细考虑过安全之后一步步发展起来的模型。

更多关于该安全模型如何根据服务器的配置来限制使用者的权限，以及恰当的suEXEC安装步骤能够避免的安全隐患，请参见警告和举例部分。

配置和安装suEXEC

继续我们的探险 ...

suEXEC配置选项

`--enable-suexec`

该选项启用默认禁止的suEXEC功能。并同时至少提供一个 `--with-suexec-xxxxx` 选项，以使APACI使用suEXEC功能来处理请求。

`--with-suexec-bin=_PATH_`

出于安全考虑， `suexec` 二进制程序的路径必须用这个选项指定并硬编码在服务器里。比如： `--with-suexec-bin=/usr/sbin/suexec`

`--with-suexec-caller=_UID_`

Apache运行时所用的UID。这是唯一允许执行程序的用户。

`--with-suexec-userdir=_DIR_`

定义suEXEC允许访问的用户宿主目录下的子目录。suEXEC将以用户身份执行这个目录下的所有可执行程序，所以这些程序必须是"安全程序"。如果使用"简单的" `UserDir` 指令(即不带"*")，则此处应该被设置为相同的值。当 *`UserDir`* 指令所指向的目录与*"passwd"*文件所指定的用户宿主目录不同时，*suEXEC*将不会正常工作，其默认值是*"public_html"*。如果所支持的虚拟主机对每个用户有不同的 *`UserDir`* ，则应该把他们集中在同一个父目录下，而用这个参数指向这个父目录。*如果配置不当，*"~userdir"*下的*cgi*请求将无效！

`--with-suexec-docroot=_DIR_`

定义Apache的 `DocumentRoot` 。它是除 `UserDir` 外suEXEC唯一可以使用的目录。其默认目录是 `--datadir` 值所指定的带有"/htdocs"的后缀的目录，比如：如果配置了" `--datadir=/home/apache` "，那么"/home/apache/htdocs"目录将作为suEXEC处理器的文档根目录。

`--with-suexec-uidmin=_UID_`

定义了suEXEC目标用户的最低UID。对大多数系统，一般是500或100。默认值是100

`--with-suexec-gidmin=_GID_`

定义了suEXEC目标组的最低GID。对大多数系统，是100，默认值也是100。

`--with-suexec-logfile=_FILE_`

它定义了suEXEC用于记录所做的事情以及发生的错误的日志文件名(对审核和排错很有用)，默认文件名是"suexec_log"，并位于标准的日志文件目录中(`--logfiledir`)。

`--with-suexec-safepath=_PATH_`

定义传给CGI程序的一个安全的PATH环境变量的值。默认值是"/usr/local/bin:/usr/bin:/bin"

编译和安装**suEXEC**处理器 若用 `--enable-suexec` 打开了suEXEC功能，那么执行 `make` 命令时(Apache自带的) `suexec` 二进制文件就会被自动建立。所有组件编译完毕后执行 `make install` 命令进行安装时， `suexec` 文件将被安装在 `--sbindir` 选项指定的目录中，默认为"/usr/local/apache2/sbin/suexec"。 注意，安装过程需要**root**权限。为了使suEXEC处理器可以设置UID，其所有者必须为 `_root_` ，并且文件模式中的执行位必须设置为1(允许执行)。

设置许可权限 虽然suEXEC包装会检查以确保它的调用者就是 `配置选项` `--with-suexec-caller` 所指定的用户。但是总是存在这样的可能性：一个系统或者库在suEXEC执行用户身份检查之前调用它，这样就存在一个可利用的漏洞。通常，避免这种问题的最佳办法是，使用文件系的统权限来确保只有Apache组用户运行的程序才能执行suEXEC。

如果你的web-server是按照如下所示进行配置的：

```
User www

Group webgroup
```

并且 `suexec` 被安装在"/usr/local/apache2/sbin/suexec"目录，你应当运行以下命令：

```
chgrp webgroup /usr/local/apache2/bin/suexec

chmod 4750 /usr/local/apache2/bin/suexec
```

这将确保只有Apache组用户运行的程序才能执行suEXEC。

启用和禁用suEXEC

Apache在启动过程中，会在 `--sbindir` 选项指定的目录(默认为："/usr/local/apache/sbin/suexec")中寻找 `suexec` 。如果Apache找到了一个正确配置的suEXEC处理器，会在错误日志中记录以下信息：

```
[notice] suEXEC mechanism enabled (wrapper: <var class="calibre40">/path/to/suexec</v
```

如果服务器启动后没有这个信息，那么很可能是服务器没找到适当的处理器，或者是这个执行程序没有安装*setuid root*。

如果要在Apache服务器运行过程中打开suEXEC功能，则必须停止并重新启动Apache。用一个简单的HUP或USR1信号来重新启动是不够的。

如果要关闭suEXEC功能，应该删除 `suexec` 文件，并停止和重新启动Apache。

使用suEXEC

对CGI程序的请求仅在下述两种情况下才会调用suEXEC包装：对一个含 `SuexecUserGroup` 指令的虚拟主机发起请求，或者该请求由 `mod_userdir` 模块处理。

虚拟主机： 使用suEXEC处理器的方法之一是在 `VirtualHost` 定义中使用 `SuexecUserGroup` 指令。通过设置这个指令来确定不同于主服务器的UID，所有对CGI资源的请求将以 `<VirtualHost>` 所定义的*User*和*Group*身份执行。如果 `<VirtualHost>` 中没有这个指令，则将以主服务器的UID身份执行。

用户目录： 由 `mod_userdir` 处理的请求会调用suEXEC处理器以被请求的用户目录所属的UID执行CGI程序。此功能的唯一要求是，此用户必须有CGI执行权限，并且其脚本符合上述安全检查的要求。参见 `--with-suexec-userdir` 编译选项。

调试suEXEC

如上所述，suEXEC处理器会在 `--with-suexec-logfile` 选项所指定的日志文件中记录信息。如果你感觉配置和安装不正常，可以查看这个日志以及服务器的错误日志。

谨防Jabberwock：警告和举例

注意！这部分文档可能还没有完成。查看最新的修订版本，请到Apache开发组的在线文档。

以下是有关限制和服务器安装的几个注意事项，在提交任何关于suEXEC的"bugs"以前，请仔细阅读。

- **suEXEC注意事项**
- 层次限制

  出于安全和效率考虑，所有suEXEC请求必须被限制在虚拟主机或者用户目录的顶层。举例来说，如果你配置了4个虚拟主机，你必须把所有虚拟主机的文档根目录都安置在同一个主Apache目录中，这样才能为虚拟主机启用suEXEC。(例子以后会有的)

- suEXEC的PATH环境变量

  改变这个变量的值是危险的，必须确保其中每个路径都是可以信任的目录。你不会希望谁都可以在你的服务器上安装特洛伊木马。

- 改变suEXEC的代码

  重申，如果你不清楚你在干什么就尽量避免，否则会带来大麻烦的。

性能方面的提示

Apache2.0是一个多用途的web服务器，其设计在灵活性、可移植性和性能中求得平衡。虽然没有在设计上刻意追求性能指标，但是Apache2.0仍然在许多现实环境中拥有很高的性能。

相比于Apache 1.3，2.0版本作了大量的优化来提升处理能力和可伸缩性，而且大多数的改进在默认状态下就可以生效。但是，在编译时和运行时，都有许多可以显著提高性能的选择。本文阐述在安装Apache2.0时，服务器管理员可以改善性能的各种方法。其中，部分配置选择可以使httpd更好地利用硬件和操作系统的兼容性，其他则是以功能换取速度。

硬件和操作系统

影响web服务器性能的最大的因素是内存。一个web服务器应该从不使用交换机制，因为交换产生的滞后使用户总感觉"不够快"，所以用户就可能去按"停止"和"刷新"，从而带来更大的负载。你可以，也应该，控制 `MaxClients` 的设置，以避免服务器产生太多的子进程而发生交换。这个过程很简单：通过 `top` 命令计算出每个Apache进程平均消耗的内存，然后再为其它进程留出足够多的内存。

其他因素就很普通了，装一个足够快的CPU，一个足够快的网卡，几个足够快的硬盘，这里说的"足够快"是指能满足实际应用的需求。

操作系统是很值得关注的又一个因素，已经被证实的很有用的经验有：

- 选择能够得到的最新最稳定的版本并打好补丁。近年来，许多操作系统厂商都提供了可以显著改善性能的TCP协议栈和线程库。
- 如果你的操作系统支持 `sendfile()` 系统调用，则务必安装带有此功能的版本或补丁(对Linux来说，就是使用Linux2.4或更高版本，对Solaris8的早期版本，则需要安装补丁)。在支持 `sendfile` 的系统中，Apache2可以更快地发送静态内容而且占用较少的CPU时间。

运行时的配置

相关模块

- `mod_dir`
- `mpm_common`
- `mod_status`

相关指令

- `AllowOverride`
- `DirectoryIndex`
- `HostnameLookups`
- `EnableMMAP`
- `EnableSendfile`
- `KeepAliveTimeout`
- `MaxSpareServers`
- `MinSpareServers`
- `Options`
- `StartServers`

HostnameLookups 和其他DNS考虑

在Apache1.3以前的版本中， `HostnameLookups` 默认被设为 `On` 。它会带来延迟，因为对每一个请求都需要作一次DNS查询。在Apache1.3中，它被默认地设置为 `Off` 。如果需要日志文件提供主机名信息以生成分析报告，则可以使用日志后处理程序 `logresolve` ，以完成DNS查询，而客户端无须等待。

推荐你最好是在其他机器上，而不是在web服务器上执行后处理和其他日志统计操作，以免影响服务器的性能。

如果你使用了任何" `Allow` from domain"或" `Deny` from domain"指令(也就是 `domain` 使用的是主机名而不是IP地址)，则代价是要进行两次DNS查询(一次正向和一次反向，以确认没有作假)。所以，为了得到最高的性能，应该避免使用这些指令(不用域名而用IP地址也是可以的)。

注意，可以把这些指令包含在 `<Location /server-status>` 段中使之局部化。在这种情况下，只有对这个区域的请求才会发生DNS查询。下例禁止除了 `.html` 和 `.cgi` 以外的所有DNS查询：

```
        HostnameLookups off

        <Files ~ "\.(html|cgi)$">

HostnameLookups on
        </Files>
```

如果在某些CGI中偶尔需要DNS名称，则可以调用 `gethostbyname` 来解决。

FollowSymLinks 和 SymLinksIfOwnerMatch

如果网站空间中没有使用 `Options FollowSymLinks` ，或使用了 `Options SymLinksIfOwnerMatch` ，Apache就必须执行额外的系统调用以验证符号连接。文件名的每一个组成部分都需要一个额外的调用。例如，如果设置了：

```
        DocumentRoot /www/htdocs

        <Directory />

Options SymLinksIfOwnerMatch
        </Directory>
```

在请求" `/index.html` "时，Apache将对" `/www` "、" `/www/htdocs` "、" `/www/htdocs/index.html` "执行 `lstat()` 调用。而且 `lstat()` 的执行结果不被缓存，因此对每一个请求都要执行一次。如果确实需要验证符号连接的安全性，则可以这样：

```
        DocumentRoot /www/htdocs

        <Directory />

Options FollowSymLinks
        </Directory>

        <Directory /www/htdocs>

Options -FollowSymLinks +SymLinksIfOwnerMatch
        </Directory>
```

这样，至少可以避免对 `DocumentRoot` 路径的多余的验证。注意，如果 `Alias` 或 `RewriteRule` 中含有 `DocumentRoot` 以外的路径，那么同样需要增加这样的段。为了得到最佳性能，应当放弃对符号连接的保护，在所有地方都设置 `FollowSymLinks` ，并放弃使用 `SymLinksIfOwnerMatch` 。

AllowOverride

如果网站空间允许覆盖(通常是用 `.htaccess` 文件)，则Apache会试图对文件名的每一个组成部分都打开 `.htaccess` ，例如：

```
        DocumentRoot /www/htdocs

        <Directory />

AllowOverride all
        </Directory>
```

如果请求" `/index.html` "，则Apache会试图打开" `/.htaccess` "、" `/www/.htaccess` "、" `/www/htdocs/.htaccess` "。其解决方法和前面所述的 `Options FollowSymLinks` 类似。为了得到最佳性能，应当对文件系统中所有的地方都使用 `AllowOverride None` 。

内容协商

实践中，内容协商的好处大于性能的损失，如果你很在意那一点点的性能损失，则可以禁止使用内容协商。但是仍然有个方法可以提高服务器的速度，就是不要使用通配符，如：

```
DirectoryIndex index
```

而使用完整的列表，如：

```
DirectoryIndex index.cgi index.pl index.shtml index.html
```

其中最常用的应该放在前面。

还有，建立一个明确的 `type-map` 文件在性能上优于使用" `Options MultiViews` "，因为所有需要的信息都在一个单独的文件中，而无须搜索目录。请参考内容协商文档以获得更详细的协商方法和创建 `type-map` 文件的指导。

内存映射

在Apache2.0需要搜索被发送文件的内容时，比如处理服务器端包含时，如果操作系统支持某种形式的 `mmap()` ，则会对此文件执行内存映射。

在某些平台上，内存映射可以提高性能，但是在某些情况下，内存映射会降低性能甚至影响到httpd的稳定性：

- 在某些操作系统中，如果增加了CPU， `mmap` 还不如 `read()` 迅速。比如，在多处理器的Solaris服务器上，关闭了 `mmap` ，Apache2.0传送服务端解析文件有时候反而更快。
- 如果你对作为NFS装载的文件系统中的一个文件进行了内存映射，而另一个NFS客户端的进程删除或者截断了这个文件，那么你的进程在下一次访问已经被映射的文件内容时，会产生一个总线错误。

如果有上述情况发生，则应该使用 `EnableMMAP off` 关闭对发送文件的内存映射。注意：此指令可以被针对目录的设置覆盖。

Sendfile

在Apache2.0能够忽略将要被发送的文件的内容的时候(比如发送静态内容)，如果操作系统支持 `sendfile()` ，则Apache将使用内核提供的 `sendfile()` 来发送文件。

在大多数平台上，使用sendfile可以通过免除分离的读和写操作来提升性能。然而在某些情况下，使用sendfile会危害到httpd的稳定性

- 一些平台可能会有Apache编译系统检测不到的有缺陷的sendfile支持，特别是将在其他平台上使用交叉编译得到的二进制文件运行于当前对sendfile支持有缺陷的平台时。

- 对于一个挂载了NFS文件系统的内核，它可能无法可靠的通过自己的cache服务于网络文件。

如果出现以上情况，你应当使用" `EnableSendfile off` "来禁用sendfile 。注意，这个指令可以被针对目录的设置覆盖。

进程的建立

在Apache1.3以前， `MinSpareServers` ， `MaxSpareServers` ， `StartServers` 的设置对性能都有很大的影响。尤其是为了应对负载而建立足够的子进程时，Apache需要有一个"渐进"的过程。在最初建立 `StartServers` 数量的子进程后，为了满足 `MinSpareServers` 设置的需要，每一秒钟只能建立一个子进程。所以，对一个需要同时处理100个客户端的服务器，如果 `StartServers` 使用默认的设置 `5` ，则为了应对负载而建立足够多的子进程需要95秒。在实际应用中，如果不频繁重新启动服务器，这样还可以，但是如果仅仅为了提供10分钟的服务，这样就很糟糕了。

"一秒钟一个"的规定是为了避免在创建子进程过程中服务器对请求的响应停顿，但是它对服务器性能的影响太大了，必须予以改变。在Apache1.3中，这个"一秒钟一个"的规定变得宽松了，创建一个进程，等待一秒钟，继续创建第二个，再等待一秒钟，继而创建四个，如此按指数级增加创建的进程数，最多达到每秒32个，直到满足 `MinSpareServers` 设置的值为止。

从多数反映看来，似乎没有必要调整 `MinSpareServers` ， `MaxSpareServers` ， `StartServers` 。如果每秒钟创建的进程数超过4个，则会在 `ErrorLog` 中产生一条消息，如果产生大量此消息，则可以考虑修改这些设置。可以使用 `mod_status` 的输出作为参考。

与进程创建相关的是由 `MaxRequestsPerChild` 引发的进程的销毁。其默认值是" `0` "，意味着每个进程所处理的请求数是不受限制的。如果此值设置得很小，比如30，则可能需要大幅增加。在SunOS或者Solaris的早期版本上，其最大值为 `10000` 以免内存泄漏。

如果启用了持久链接，子进程将保持忙碌状态以等待被打开连接上的新请求。为了最小化其负面影响， `KeepAliveTimeout` 的默认值被设置为 `5` 秒，以谋求网络带宽和服务器资源之间的平衡。在任何情况下此值都不应当大于 `60` 秒，参见most of the benefits are lost。

编译时的配置

选择一个MPM

Apache 2.x 支持插入式并行处理模块，称为多路处理模块(MPM)。在编译Apache时你必须选择也只能选择一个MPM，这里有几个针对非UNIX系统的MPM： `beos` ， `mpm_netware` ， `mpmt_os2` ， `mpm_winnt` 。对类UNIX系统，有几个不同的MPM可供选择，他们都会影响到httpd的速度和可伸缩性：

- `worker` MPM使用多个子进程，每个子进程中又有多个线程。每个线程处理一个请求。该

MPM通常对高流量的服务器是一个不错的选择。因为它比 `prefork` MPM需要更少的内存且更具有伸缩性。
- `prefork` MPM使用多个子进程，但每个子进程并不包含多线程。每个进程只处理一个链接。在许多系统上它的速度和 `worker` MPM一样快，但是需要更多的内存。这种无线程的设计在某些情况下优于 `worker` MPM：它可以应用于不具备线程安全的第三方模块(比如php3/4/5)，且在不支持线程调试的平台上易于调试，而且还具有比 `worker` MPM更高的稳定性。

关于MPM的更多内容，请参考其文档。

模块

既然内存用量是影响性能的重要因素，你就应当尽量去除你不需要的模块。如果你将模块编译成DSO，取消不必要的模块就是一件非常简单的事情：注释掉 `LoadModule` 指令中不需要的模块。

如果你已经将模块静态链接进Apache二进制核心，你就必须重新编译Apache并去掉你不想要的模块。

增减模块牵涉到的一个问题是：究竟需要哪些模块、不需要哪些模块？这取决于服务器的具体情况。一般说来，至少要包含下列模块： `mod_mime` ， `mod_dir` ， `mod_log_config` 。你也可以不要 `mod_log_config` ，但是一般不推荐这样做。

原子操作

一些模块，比如 `mod_cache` 和 `worker` 使用APR(Apache可移植运行时)的原子API。这些API提供了能够用于轻量级线程同步的原子操作。

默认情况下，APR在每个目标OS/CPU上使用其最有效的特性执行这些操作。比如许多现代CPU的指令集中有一个原子的比较交换(compare-and-swap, CAS)操作指令。在一些老式平台上，APR默认使用一种缓慢的、基于互斥执行的原子API以保持对没有CAS指令的老式CPU的兼容。如果你只打算在新式的CPU上运行Apache，你可以在编译时使用 `--enable-nonportable-atomics` 选项：

```
./buildconf
./configure --with-mpm=worker --enable-nonportable-atomics=yes
```

`--enable-nonportable-atomics` 选项只和下列平台相关：

- SPARC上的Solaris 默认情况下，APR使用基于互斥执行的原子操作。如果你使用 `--enable-nonportable-atomics` 选项，APR将使用SPARC v8plus操作码来加快基于硬件的CAS操作。注意，这仅对UltraSPARC CPU有效。
- x86上的Linux 默认情况下，APR在Linux上使用基于互斥执行的原子操作。如果你使用

`--enable-nonportable-atomics` 选项，APR将使用486操作码来加快基于硬件的CAS操作。注意，这仅对486以上的CPU有效。

mod_status 和 "ExtendedStatus On"

如果Apache在编译时包含了 `mod_status` ，而且在运行时设置了" `ExtendedStatus On` "，那么Apache会对每个请求调用两次 `gettimeofday()` (或者根据操作系统的不同，调用 `times()`)以及(1.3版之前)几个额外的 `time()` 调用，使状态记录带有时间标志。为了得到最佳性能，可以设置" `ExtendedStatus off` "(这也是默认值)。

多socket情况下的串行accept

警告

这部分内容尚未完全根据Apache2.0中的变化进行更新 。一些信息依然有效，使用中请注意。

这里要说的是 Unix socket API 的一个缺点。假设web服务器使用了多个 `Listen` 语句监听多个端口或者多个地址，Apache会使用 `select()` 以检测每个socket是否就绪。 `select()` 会表明一个socket有零或至少一个连接正等候处理。由于Apache的模型是多子进程的，所有空闲进程会同时检测新的连接。一个很天真的实现方法是这样的(这些例子并不是源代码，只是为了说明问题而已)：

```
        for (;;) {
  for (;;) {
  fd_set accept_fds;
            FD_ZERO (&accept_fds);
            for (i = first_socket; i <= last_socket; ++i) {
  FD_SET (i, &accept_fds);
            }
            rc = select (last_socket+1, &accept_fds, NULL, NULL, NULL);
            if (rc < 1) continue;
            new_connection = -1;
            for (i = first_socket; i <= last_socket; ++i) {
  if (FD_ISSET (i, &accept_fds)) {
  new_connection = accept (i, NULL, NULL);
                if (new_connection != -1) break;
              }
            }
            if (new_connection != -1) break;
          }
          process the new_connection;
        }
```

这种天真的实现方法有一个严重的"饥饿"问题。如果多个子进程同时执行这个循环，则在多个请求之间，进程会被阻塞在 `select` ，随即进入循环并试图 `accept` 此连接，但是只有一个进程可以成功执行(假设还有一个连接就绪)，而其余的则会被阻塞在 `accept` 。这样，只有那一个socket可以处理请求，而其他都被锁住了，直到有足够多的请求将它们唤醒。此"饥饿"问题在PR#467中有专门的讲述。目前至少有两种解决方案。

一种方案是使用非阻塞型socket，不阻塞子进程并允许它们立即继续执行。但是这样会浪费CPU时间。设想一下， `select` 有10个子进程，当一个请求到达的时候，其中9个被唤醒，并试图 `accept` 此连接，继而进入 `select` 循环，无所事事，并且其间没有一个子进程能够响应出现在其他socket上的请求，直到退出 `select` 循环。总之，这个方案效率并不怎么高，除非你有很多的CPU，而且开了很多子进程。

另一种也是Apache所使用的方案是，使内层循环的入口串行化，形如(不同之处以高亮显示)：

```
        for (;;) {
  **accept_mutex_on ();**
          for (;;) {
  fd_set accept_fds;
            FD_ZERO (&accept_fds);
            for (i = first_socket; i <= last_socket; ++i) {
  FD_SET (i, &accept_fds);
            }
            rc = select (last_socket+1, &accept_fds, NULL, NULL, NULL);
            if (rc < 1) continue;
            new_connection = -1;
            for (i = first_socket; i <= last_socket; ++i) {
  if (FD_ISSET (i, &accept_fds)) {
  new_connection = accept (i, NULL, NULL);
                if (new_connection != -1) break;
              }
            }
            if (new_connection != -1) break;
          }
 **accept_mutex_off ();**
          process the new_connection;
        }
```

函数 `accept_mutex_on` 和 `accept_mutex_off` 实现了一个互斥信号灯，在任何时刻只被为一个子进程所拥有。实现互斥的方法有多种，其定义位于 `src/conf.h` (1.3以前的版本)或 `src/include/ap_config.h` (1.3或以后的版本)中。在一些根本没有锁定机制的体系中，使用多个 `Listen` 指令就是不安全的。

`AcceptMutex` 指令被用来改变在运行时使用的互斥方案。

`AcceptMutex flock`

这种方法调用系统函数 `flock()` 来锁定一个加锁文件(其位置取决于 `LockFile` 指令)。

`AcceptMutex fcntl`

这种方法调用系统函数 `fcntl()` 来锁定一个加锁文件(其位置取决于 `LockFile` 指令)。

`AcceptMutex sysvsem`

(1.3及更新版本)这种方案使用SysV风格的信号灯以实现互斥。不幸的是，SysV风格的信号灯有一些副作用，其一是，Apache有可能不能在结束以前释放这种信号灯(见 `ipcs()` 的man page)，另外，这种信号灯API给与网络服务器有相同uid的CGI提供了拒绝服务攻击的机会(所

有CGI，除非用了类似 `suexec` 或 `cgiwrapper`)。鉴于此，在多数体系中都不用这种方法，除了IRIX(因为前两种方法在IRIX中代价太高)。

`AcceptMutex pthread`

(1.3及更新版本)这种方法使用了POSIX互斥，按理应该可以用于所有完整实现了POSIX线程规范的体系中，但是似乎只能用在Solaris2.5及更新版本中，甚至只能在某种配置下才正常运作。如果遇到这种情况，则应该提防服务器的挂起和失去响应。只提供静态内容的服务器可能不受影响。

`AcceptMutex posixsem`

(2.0及更新版本)这种方法使用了POSIX信号灯。如果一个运行中的线程占有了互斥segfault，则信号灯的所有者将不会被恢复，从而导致服务器的挂起和失去响应。

如果你的系统提供了上述方法以外的串行机制，那就可能需要为APR增加代码(或者提交一个补丁给Apache)。

还有一种曾经考虑过但从未予以实施的方案是使循环部分地串行化，即只允许一定数量的进程进入循环。这种方法仅在多个进程可以同时进行的多处理器的系统中才是有价值的，而且这样的串行方法并没有占用整个带宽。它也许是将来研究的一个领域，但是由于高度并行的网络服务器并不符合规范，所以其被优先考虑的程度会比较低。

当然，为了得到最佳性能，最后就根本不使用多个 `Listen` 语句。但是上述内容还是值得读一读。

单socket情况下的串行accept

上述对多socket的服务器进行了一流的讲述，那么对单socket的服务器又怎样呢？理论上似乎应该没有什么问题，因为所有进程在连接到来的时候可以由 `accept()` 阻塞，而不会产生进程"饥饿"的问题，但是在实际应用中，它掩盖了与上述非阻塞方案几乎相同的问题。按大多数TCP栈的实现方法，在单个连接到来时，内核实际上唤醒了所有阻塞在 `accept` 的进程，但只有一个能得到此连接并返回到用户空间，而其余的由于得不到连接而在内核中处于休眠状态。这种休眠状态为代码所掩盖，但的确存在，并产生与多socket中采用非阻塞方案相同的负载尖峰的浪费。

同时，我们发现在许多体系结构中，即使在单socket的情况下，实施串行化的效果也不错，因此在几乎所有的情况下，事实上就都这样处理了。在Linux(2.0.30，双Pentium pro 166/128M RAM)下的测试显示，对单socket，串行化比不串行化每秒钟可以处理的请求少了不到3%，但是，不串行化对每一个请求多了额外的100ms的延迟，此延迟可能是因为长距离的网络线路所致，并且仅发生在LAN中。如果需要改变对单socket的串行化，可以定义 `SINGLE_LISTEN_UNSERIALIZED_ACCEPT` ，使单socket的服务器彻底放弃串行化。

延迟的关闭

正如draft-ietf-http-connection-00.txt section 8所述，HTTP服务器为了可靠地实现此协议，需要单独地在每个方向上关闭通讯(重申一下，一个TCP连接是双向的，两个方向之间是独立的)。在这一点上，其他服务器经常敷衍了事，但从1.2版本开始被Apache正确实现了。

但是增加了此功能以后，由于一些Unix版本的短见，随之也出现了许多问题。TCP规范并没有规定 `FIN_WAIT_2` 必须有一个超时，但也没有明确禁止。在没有超时的系统中，Apache1.2经常会陷于 `FIN_WAIT_2` 状态中。多数情况下，这个问题可以用供应商提供的TCP/IP补丁予以解决。而如果供应商不提供补丁(指SunOS4 -- 尽管用户们持有允许自己修补代码的许可证)，那么只能关闭此功能。

实现的方法有两种，其一是socket选项 `SO_LINGER` ，但是似乎命中注定，大多数TCP/IP栈都从未予以正确实现。即使在正确实现的栈中(指Linux2.0.31)，此方法也被证明其代价比下一种方法高昂。

Apache对此的实现代码大多位于函数 `lingering_close` (位于 `http_main.c`)中。此函数大致形如：

```
        void lingering_close (int s)
        {
   char junk_buffer[2048];
          /* shutdown the sending side */
          shutdown (s, 1);
          signal (SIGALRM, lingering_death);
          alarm (30);
          for (;;) {
   select (s for reading, 2 second timeout);
            if (error) break;
            if (s is ready for reading) {
   if (read (s, junk_buffer, sizeof (junk_buffer)) <= 0) {
   break;
              }
              /* just toss away whatever is here */
            }
          }
          close (s);
        }
```

此代码在连接结束时多了一些开销，但这是可靠实现所必须的。由于HTTP/1.1越来越流行，而且所有连接都是稳定的，此开销将由更多的请求共同分担。如果你要玩火去关闭这个功能，可以定义 `NO_LINGCLOSE` ，但绝不推荐这样做。尤其是，随着HTTP/1.1中管道化稳定连接的启用， `lingering_close` 已经成为绝对必须。而且，管道化连接速度更快，应该考虑予以支持。

Scoreboard 文件

Apache父进程和子进程通过scoreboard进行通讯。通过共享内存来实现当然是最理想的。在我们曾经实践过或者提供了完整移植的操作系统中，都使用共享内存，其余的则使用磁盘文件。磁盘文件不仅速度慢，而且不可靠(功能也少)。仔细阅读你的体系所对应的 `src/main/conf.h` 文件，并查找 `USE_MMAP_SCOREBOARD` 或 `USE_SHMGET_SCOREBOARD` 。定义其中之一(或者分别类似HAVE_MMAP和HAVE_SHMGET)，可以使共享内容的相关代码生效。如果你的系统提供其他类型的共享内容，则需要修改 `src/main/http_main.c` 文件，并把必需的挂钩添加到服务器中。(也请发送一个补丁给我们)

注意：在对Linux的Apache1.2移植版本之前，没有使用内存共享，此失误使Apache的早期版本在Linux中表现很差。

DYNAMIC_MODULE_LIMIT

如果你不想使用动态加载模块(或者是因为看见了这段话，或者是为了获得最后一点点性能上的提高)，可以在编译服务器时定义 `-DDYNAMIC_MODULE_LIMIT=0` ，这样可以节省为支持动态加载模块而分配的内存。

附录：踪迹的详细分析

在Solaris8的MPM中，Apache2.0.38使用一个系统调用以收集踪迹：

```
truss -l -p <var class="calibre40">httpd_child_pid</var>.
```

`-l` 参数使truss记录每个执行系统调用的LWP(lightweight process--Solaris核心级线程)的ID。

其他系统可能使用不同的系统调用追踪工具，诸如 `strace` , `ktrace` , `par` ，其输出都是相似的。

下例中，一个客户端向httpd请求了一个10KB的静态文件。对非静态或内容协商请求的记录会有很大不同(有时也很难看明白)。

```
/67:    accept(3, 0x00200BEC, 0x00200C0C, 1) (sleeping...)
/67:    accept(3, 0x00200BEC, 0x00200C0C, 1)            = 9
```

下例中，监听线程是 LWP #67 。

注意对 `accept()` 串行化支持的匮乏。与这个特殊平台对应的MPM在默认情况下使用非串行的accept，除了在监听多个端口的时候。

```
/65:    lwp_park(0x00000000, 0)                          = 0
/67:    lwp_unpark(65, 1)                                = 0
```

接受了一个连接后，监听线程唤醒一个工作线程以处理此请求。下例中，处理请求的那个工作线程是 LWP #65 。

```
/65:    getsockname(9, 0x00200BA4, 0x00200BC4, 1)       = 0
```

为了实现虚拟主机，Apache需要知道接受连接的本地socket地址。在许多情况下，有可能无须执行此调用(比如没有虚拟主机，或者 `Listen` 指令中没有使用通配地址)，但是目前并没有对此作优化处理。

```
/65:    brk(0x002170E8)                                  = 0
/65:    brk(0x002190E8)                                  = 0
```

此 `brk()` 调用是从堆中分配内存的，它在系统调用记录中并不多见，因为httpd在多数请求处理中使用了自己的内存分配器(`apr_pool` 和 `apr_bucket_alloc`)。下例中，httpd刚刚启动，所以它必须调用 `malloc()` 以分配原始内存块用于自己的内存分配器。

```
/65:    fcntl(9, F_GETFL, 0x00000000)                    = 2
/65:    fstat64(9, 0xFAF7B818)                           = 0
/65:    getsockopt(9, 65535, 8192, 0xFAF7B918, 0xFAF7B910, 2190656) = 0
/65:    fstat64(9, 0xFAF7B818)                           = 0
/65:    getsockopt(9, 65535, 8192, 0xFAF7B918, 0xFAF7B914, 2190656) = 0
/65:    setsockopt(9, 65535, 8192, 0xFAF7B918, 4, 2190656) = 0
/65:    fcntl(9, F_SETFL, 0x00000082)                    = 0
```

接着，工作线程使客户端连接处于非阻塞模式。 `setsockopt()` 和 `getsockopt()` 调用是Solaris的libc对socket执行 `fcntl()` 所必须的。

```
/65:    read(9, " G E T   / 1 0 k . h t m".., 8000)     = 97
```

工作线程从客户端读取请求。

```
/65:    stat("/var/httpd/apache/httpd-8999/htdocs/10k.html", 0xFAF7B978) = 0
/65:    open("/var/httpd/apache/httpd-8999/htdocs/10k.html", O_RDONLY) = 10
```

这里，httpd被配置为" `Options FollowSymLinks` "和" `AllowOverride None` "。所以，无须对每个被请求文件路径中的目录执行 `lstat()` ，也不需要检查 `.htaccess` 文件，它简单地调用 `stat()` 以检查此文件是否存在，以及是一个普通的文件还是一个目录。

```
/65:    sendfilev(0, 9, 0x00200F90, 2, 0xFAF7B53C)      = 10269
```

此例中，httpd可以通过单个系统调用 `sendfilev()` 发送HTTP响应头和被请求的文件。Sendfile因操作系统会有所不同，有些系统中，在调用 `sendfile()` 以前，需要调用 `write()` 或 `writev()` 以发送响应头。

```
/65:    write(4, " 1 2 7 . 0 . 0 . 1   -  ".., 78)      = 78
```

此 `write()` 调用在访问日志中对请求作了记录。注意，其中没有对 `time()` 的调用的记录。与Apache1.3不同，Apache2.0使用 `gettimeofday()` 以查询时间。在有些操作系统中，比如Linux和Solaris， `gettimeofday` 有一个优化的版本，其开销比一个普通的系统调用要小一点。

```
/65:    shutdown(9, 1, 1)                               = 0
/65:    poll(0xFAF7B980, 1, 2000)                       = 1
/65:    read(9, 0xFAF7BC20, 512)                        = 0
/65:    close(9)                                        = 0
```

工作线程对连接作延迟的关闭。

```
/65:    close(10)                                       = 0
/65:    lwp_park(0x00000000, 0)         (sleeping...)
```

最后，工作线程关闭发送完的文件和块，直到监听进程把它指派给另一个连接。

```
/67:    accept(3, 0x001FEB74, 0x001FEB94, 1) (sleeping...)
```

其间，监听进程可以在把一个连接指派给一个工作进程后立即接受另一个连接(但是如果所有工作进程都处于忙碌状态，则会受MPM中的一些溢出控制逻辑的制约)。虽然在此例中并不明显，在工作线程刚接受了一个连接之后，下一个 `accept()` 会(在高负荷的情况下更会)立即并行产生。

URL重写指南

Originally written by <cite class="calibre27">Ralf S. Engelschall <rse@apache.org></cite>
December 1997

本文是 `mod_rewrite` 的参考文档，阐述在实际应用中如何解决网管所面临的基于URL的典型问题，并详细描述了如何配置URL重写规则集以解决问题。

`mod_rewrite` 简介

Apache的 `mod_rewrite` 是提供了强大URL操作的杀手级的模块，可以实现几乎所有你梦想的URL操作类型，其代价是你必须接受其复杂性，因为 `mod_rewrite` 的主要障碍就是初学者不容易理解和运用，即使是Apache专家有时也会发掘出 `mod_rewrite` 的新用途。换句话说：对于 `mod_rewrite` ，或者是打退堂鼓永不再用，或者是喜欢它并一生受用。本文试图通过对已有方案的表述来创造一个成功的开端，以免你放弃。

实践方案

我自己创造和收集了许多实践方案，不要有畏惧心理，从这些例子开始学习URL重写的黑匣子吧。

注意：根据你的服务器配置，可能有必要对例子作些微修改，比如，新启用 `mod_alias` 和 `mod_userdir` 时要增加[PT]标志，或者重写.htaccess而不是单个服务器中的规则集。对一个特定的规则集应该先透彻理解然后再考虑应用，这样才能避免出现问题。

URL 的规划

规范的URL

说明：

在有些web服务器上，一个资源会拥有多个URL，在实际应用和发布中应该被使用的是规范的URL，其他的则是简写或者只在内部使用。无论用户在请求中使用什么形式的URL，他最终看见的都应该是规范的URL。

方案：

对所有不规范的URL执行一个外部HTTP重定向，以改变它在浏览器地址栏中的显示及其后继请求。下例中的规则集用规范的/u/user替换/~user，并修正了/u/user所遗漏的后缀斜杠。

```
RewriteRule   ^/**~**([^/]+)/?(.*)    /**u**/$1/$2  [**R**]
RewriteRule   ^/([uge])/(**[^/]+**)$  /$1/$2**/**   [**R**]
```

规范的主机名

说明：

这个规则的目的是强制使用特定的主机名以代替其他名字。比如，你想强制使用 www.example.com 代替 example.com ，就可以在以下方法的基础上进行修改：

方案：

```
# 针对运行在非80端口的站点
RewriteCond %{HTTP_HOST}   !^fully\.qualified\.domain\.name [NC]
RewriteCond %{HTTP_HOST}   !^$
RewriteCond %{SERVER_PORT} !^80$
RewriteRule ^/(.*)         http://fully.qualified.domain.name:%{SERVER_PORT}/$1 [L,R]

# 对一个运行在80端口的站点
RewriteCond %{HTTP_HOST}   !^fully\.qualified\.domain\.name [NC]
RewriteCond %{HTTP_HOST}   !^$
RewriteRule ^/(.*)         http://fully.qualified.domain.name/$1 [L,R]
```

移动过的 `DocumentRoot`

说明：

通常，web服务器的 `DocumentRoot` 直接对应于URL"/"，但是，它常常不是处于最高一级，而可能只是众多数据池中的一个实体。比如，在Intranet站点中，有/e/www/(WWW的主页)、/e/sww/(Intranet的主页)等等，而 `DocumentRoot` 指向了/e/www/，则必须保证此数据池中所有内嵌的图片和其他元素对后继请求有效。

方案：

只须重定向URL"/"到"/e/www/"即可。这个方案看起来很简单，但只是因为有了mod_rewrite模块的支持，它才简单，因为传统的URL Aliases机制(由mod_alias及其相关模块提供)只是作了一个前缀匹配，DocumentRoot是一个对所有URL的前缀，因而无法实现这样的重定向。而用mod_rewrite的确很简单：

```
RewriteEngine on
RewriteRule   **^/$**  /e/www/  [**R**]
```

注意， 也可以通过 `RedirectMatch` 指令达到这个目的：

```
    RedirectMatch ^/$ http://example.com/e/www/
```

后缀斜杠的问题

说明：

每个网管对引用目录后缀斜杠的问题都有一本苦经，如果遗漏了，服务器会产生一个错误，因为如果请求是/~quux/foo而不是/~quux/foo/，服务器就会去找一个叫foo的文件，而它是一个目录，所以就报错了。事实上，大多数情况下，它自己会试图修正这个错误，但是有时候需要你手工纠正，比如，在重写了许多CGI脚本中的复杂的URL以后。

方案：

解决这个微妙问题的方案是让服务器自动添加后缀斜杠。对此，必须使用一个外部重定向，使浏览器正确地处理后继的对诸如图片的请求。如果仅仅作一个内部重写，可能只对目录页面有效，而对内嵌有使用相对URL的图片的页面无效，因为浏览器有请求内嵌目标的可能。比如，如果不用外部重定向，/~quux/foo/index.html页面中对image.gif的请求，其结果将是/~quux/image.gif

所以，应该这样写：

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^foo**$**  foo**/**  [**R**]
```

又懒又疯狂的做法是把这些写入其宿主目录中的顶级.htaccess中，但是须注意，如此会带来一些处理上的开销。

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteCond    %{REQUEST_FILENAME}  **-d**
RewriteRule    ^(.+**[^/]**)$           $1**/**  [R]
```

集群网站的同类URL规划

说明：

我们希望在一个Intranet集群网站中，对所有WWW服务器建立一致的URL规划，也就是说，所有的URL(针对每个服务器进行本地配置，因此是独立于各个服务器的)实际上都是独立于各个服务器的！我们需要的是一个具有独立于各个服务器的一致性规划的WWW名称空间，即URL不需要包含物理目标服务器，而由集群本身来自动定位物理目标主机。

方案：

首先，目标服务器的信息来自(产生)于包含有用户、组以及实体的外部地图，其格式形如：

```
user1  server_of_user1
user2  server_of_user2
:      :
```

这些信息被存入map.xxx-to-host文件。其次，如果URL在一个服务器上无效，需要引导所有的服务器重定向URL

```
/u/user/anypath
/g/group/anypath
/e/entity/anypath
```

到

```
http://physical-host/u/user/anypath
http://physical-host/g/group/anypath
http://physical-host/e/entity/anypath
```

以下规则集依靠映射文件来完成这个操作(假定，如果一个用户在映射中没有对应的项，则使用server0为默认服务器)：

```
RewriteEngine on

RewriteMap      user-to-host   txt:/path/to/map.user-to-host
RewriteMap     group-to-host   txt:/path/to/map.group-to-host
RewriteMap    entity-to-host   txt:/path/to/map.entity-to-host

RewriteRule   ^/u/**([^/]+)**/?(.*)   http://**${user-to-host:$1|server0}**/u/$1/$2
RewriteRule   ^/g/**([^/]+)**/?(.*)  http://**${group-to-host:$1|server0}**/g/$1/$2
RewriteRule   ^/e/**([^/]+)**/?(.*) http://**${entity-to-host:$1|server0}**/e/$1/$2

RewriteRule   ^/([uge])/([^/]+)/?$          /$1/$2/.www/
RewriteRule   ^/([uge])/([^/]+)/([^.]+.+)   /$1/$2/.www/$3\
```

移动用户主目录到不同的web服务器

说明：

通常，许多网管在建立一个新的web服务器时，都会有这样的要求：重定向一个web服务器上的所有用户主目录到另一个web服务器。

方案：

很简单，在老的web服务器上重定向所有的URL"/~user/anypath"到http://newserver/~user/anypath

```
RewriteEngine on
RewriteRule   ^/~(.+)  http://**newserver**/~$1  [R,L]
```

结构化的用户主目录

说明：

一些拥有几千个用户的网站通常都使用结构化的用户主目录规划，即每个用户主目录位于一个带有特定前缀，比如其用户名的第一个字符的子目录下：/~foo/anypath代表/home/**f**/foo/.www/anypath，而/~bar/anypath代表/home/**b**/bar/.www/anypath

方案：

可以使用下列规则集来扩展~以达到上述目的。

```
RewriteEngine on
RewriteRule   ^/~(**([a-z])**[a-z0-9]+)(.*)  /home/**$2**/$1/.www$3
```

文件系统的重组

说明：

这是一个不加雕琢的例子：一个大量使用针对目录的规则集以实现平滑的观感，并且从来不用调整数据结构的杀手级的应用。背景：net.sw从1992年开始，存放了我收集的免费Unix软件包。它是我的爱好也是我的工作，因为在学习计算机科学的同时，业余时间还做了多年的系统和网络管理员。每周我都需要整理软件，因而建立了一个层次很深的目录结构来存放各种软件包：

```
drwxrwxr-x   2 netsw  users    512 Aug  3 18:39 Audio/
drwxrwxr-x   2 netsw  users    512 Jul  9 14:37 Benchmark/
drwxrwxr-x  12 netsw  users    512 Jul  9 00:34 Crypto/
drwxrwxr-x   5 netsw  users    512 Jul  9 00:41 Database/
drwxrwxr-x   4 netsw  users    512 Jul 30 19:25 Dicts/
drwxrwxr-x  10 netsw  users    512 Jul  9 01:54 Graphic/
drwxrwxr-x   5 netsw  users    512 Jul  9 01:58 Hackers/
drwxrwxr-x   8 netsw  users    512 Jul  9 03:19 InfoSys/
drwxrwxr-x   3 netsw  users    512 Jul  9 03:21 Math/
drwxrwxr-x   3 netsw  users    512 Jul  9 03:24 Misc/
drwxrwxr-x   9 netsw  users    512 Aug  1 16:33 Network/
drwxrwxr-x   2 netsw  users    512 Jul  9 05:53 Office/
drwxrwxr-x   7 netsw  users    512 Jul  9 09:24 SoftEng/
drwxrwxr-x   7 netsw  users    512 Jul  9 12:17 System/
drwxrwxr-x  12 netsw  users    512 Aug  3 20:15 Typesetting/
drwxrwxr-x  10 netsw  users    512 Jul  9 14:08 X11/
```

1996年7月，我决定通过一个漂亮的Web接口公开我的收藏。"漂亮"是指提供一个接口以直接浏览整个目录结构，同时不对这个结构做任何改变，甚至也不在结构顶部放置CGI脚本。为什么呢？因为这个结构还要能够被FTP访问，而且我不希望其中有任何Web或者CGI成分。

方案：

这个方案分为两个部分：第一个部分，是用于在空闲时间建立所有目录页面的CGI脚本集。我把它们放在/e/netsw/.www/，如下：

```
-rw-r--r--   1 netsw  users    1318 Aug  1 18:10 .wwwacl
drwxr-xr-x  18 netsw  users     512 Aug  5 15:51 DATA/
-rw-rw-rw-   1 netsw  users  372982 Aug  5 16:35 LOGFILE
-rw-r--r--   1 netsw  users     659 Aug  4 09:27 TODO
-rw-r--r--   1 netsw  users    5697 Aug  1 18:01 netsw-about.html
-rwxr-xr-x   1 netsw  users     579 Aug  2 10:33 netsw-access.pl
-rwxr-xr-x   1 netsw  users    1532 Aug  1 17:35 netsw-changes.cgi
-rwxr-xr-x   1 netsw  users    2866 Aug  5 14:49 netsw-home.cgi
drwxr-xr-x   2 netsw  users     512 Jul  8 23:47 netsw-img/
-rwxr-xr-x   1 netsw  users   24050 Aug  5 15:49 netsw-lsdir.cgi
-rwxr-xr-x   1 netsw  users    1589 Aug  3 18:43 netsw-search.cgi
-rwxr-xr-x   1 netsw  users    1885 Aug  1 17:41 netsw-tree.cgi
-rw-r--r--   1 netsw  users     234 Jul 30 16:35 netsw-unlimit.lst
```

其中的"DATA"子目录包含了上述目录结构，即实在的net.sw ，由rdist在需要的时候自动更新。第二个部分的遗留问题是：如何连接这两个结构为一个平滑观感的URL树？我希望在运行适当的CGI脚本而使用各种URL的时候，使用户感觉不到"DATA"目录的存在。方案如下：首先，我把下列配置放在服务器上DocumentRoot中针对目录的配置文件里，重写公布的URL"/net.sw/"为内部路径"/e/netsw" ：

```
RewriteRule  ^net.sw$       net.sw/        [R]
RewriteRule  ^net.sw/(.*)$  e/netsw/$1
```

第一条规则是针对遗漏后缀斜杠的请求的！第二条规则才是真正实现功能的。接着，就是放在针对目录的配置文件/e/netsw/.www/.wwwacl中的杀手级的配置了：

```
Options       ExecCGI FollowSymLinks Includes MultiViews

RewriteEngine on

# 我们通过"/net.sw/"前缀到达
RewriteBase   /net.sw/

# 首先重写根目录到cgi处理脚本
RewriteRule   ^$                       netsw-home.cgi     [L]
RewriteRule   ^index\.html$            netsw-home.cgi     [L]

#  当浏览器请求perdir页面时剥去子目录
RewriteRule   ^.+/(netsw-[^/]+/.+)$    $1                 [L]

# 现在打断本地文件的重写
RewriteRule   ^netsw-home\.cgi.*       -                  [L]
RewriteRule   ^netsw-changes\.cgi.*    -                  [L]
RewriteRule   ^netsw-search\.cgi.*     -                  [L]
RewriteRule   ^netsw-tree\.cgi$        -                  [L]
RewriteRule   ^netsw-about\.html$      -                  [L]
RewriteRule   ^netsw-img/.*$           -                  [L]

# 任何别的东西都是一个由另一个cgi脚本处理的子目录
RewriteRule   !^netsw-lsdir\.cgi.*     -                  [C]
RewriteRule   (.*)                     netsw-lsdir.cgi/$1
```

阅读提示：

1. 注意前半部分中的标志L(最后)，和无对应项("-")
2. 注意后半部分中的符号!(非)，和标志C(链)
3. 注意最后一条规则的全匹配模式

NCSA图像映射和 `mod_imap`

说明：

许多人都希望在从NCSA web服务器向较现代的Apache web服务器转移中实现平滑过渡，即希望老的NCSA图像映射程序能在Apache的较现代的 `mod_imagemap` 支持下正常运作。但问题在于，到处都是通过/cgi-bin/imagemap/path/to/page.map引用imagemap程序的连接，而在Apache下，应该写成/path/to/page.map

方案：

使用全局规则在传输过程中去除所有这些请求的前缀：

```
RewriteEngine  on
RewriteRule    ^/cgi-bin/imagemap(.*)  $1  [PT]
```

在多个目录中搜索页面

说明：

有时会有必要使web服务器在多个目录中搜索页面，对此，MultiViews或者其他技术无能为力。

方案：

编制一个明确的规则集以搜索目录中的文件。

```
RewriteEngine on

# 首先尝试在 custom/...中寻找
RewriteCond         /your/docroot/**dir1**/%{REQUEST_FILENAME}  -f
RewriteRule  ^(.+)  /your/docroot/**dir1**/$1  [L]

# 然后尝试在 pub/...中寻找
RewriteCond         /your/docroot/**dir2**/%{REQUEST_FILENAME}  -f
RewriteRule  ^(.+)  /your/docroot/**dir2**/$1  [L]

# 再找不到就继续寻找其他的Alias 或 ScriptAlias 目录...
RewriteRule   ^(.+)  -  [PT]
```

按照URL的片段设置环境变量

说明：

如果希望保持请求之间的状态信息，又不希望使用CGI来包装所有页面，而是只通过分离URL中的有用信息来编码。

方案：

可以用一个规则集来分离出状态信息，并设置环境变量以备此后用于XSSI或CGI 。这样，一个"/foo/S=java/bar/"的URL会被解析为/foo/bar/ ，而环境变量STATUS则被设置为"java"。

```
RewriteEngine on
RewriteRule   ^(.*)/**S=([^/]+)**/(.*)    $1/$3 [E=**STATUS:$2**]
```

虚拟用户主机

说明：

如果需要为用户**username**支持一个www.**username**.host.domain.com的主页，但不是用在此机器上建虚拟主机的方法，而是用仅在此机器上增加一个DNS记录的方法实现。

方案：

对HTTP/1.0的请求，这是无法实现的；但是对HTTP/1.1的在HTTP头中包含有主机名的请求，可以用以下规则集来内部地重写http://www.username.host.com/anypath 为/home/username/anypath

```
RewriteEngine on
RewriteCond   %{**HTTP_HOST**}                 ^www\.**[^.]+**\.host\.com$
RewriteRule   ^(.+)                        %{HTTP_HOST}$1          [C]
RewriteRule   ^www\.**([^.]+)**\.host\.com(.*) /home/**$1**$2
```

为外来访问者重定向用户主目录

说明：

对不是来自本地域ourdomain.com的外来访问者的请求，重定向其用户主目录URL到另一个web服务器www.somewhere.com ，有时这种做法也会用在虚拟主机的配置段中。

方案：

只须一个重写条件：

```
RewriteEngine on
RewriteCond   %{REMOTE_HOST}  **!^.+\.ourdomain\.com$**
RewriteRule   ^(/~.+)         http://www.somewhere.com/$1 [R,L]
```

重定向失败的URL到其他web服务器

说明：

如何重写URL以重定向对web服务器A的失败请求到服务器B，是一个常见的问题。一般，可以用Perl写的CGI脚本通过 `ErrorDocument` 来解决，此外，还有 `mod_rewrite` 方案。但是须注意，这种方法的执行效率不如用 `ErrorDocument` 的CGI脚本!

方案：

第一种方案，有最好的性能而灵活性欠佳，出错概率小所以安全：

```
RewriteEngine on
RewriteCond   /your/docroot/%{REQUEST_FILENAME} **!-f**
RewriteRule   ^(.+)                             http://**webserverB**.dom/$1
```

但是其问题在于，它只对位于 `DocumentRoot` 中的页面有效。虽然可以增加更多的条件(比如同时还处理用户主目录，等等)，但是还有一个更好的方法：

```
RewriteEngine on
RewriteCond   %{REQUEST_URI} **!-U**
RewriteRule   ^(.+)          http://**webserverB**.dom/$1
```

这种方法使用了 `mod_rewrite` 提供的"向前参照"(look-ahead)的功能，是一种对所有URL类型都有效而且安全的方法。但是，对web服务器的性能会有影响，所以如果web服务器有一个强大的CPU，那就用这个方法。而在慢速机器上，可以用第一种方法，或者用性能更好的 `ErrorDocument` CGI脚本。

扩展的重定向

说明：

有时候，我们会需要更多的对重定向URL的(有关字符转义机制方面的)控制。通常，Apache内核中的URL转义函数uri_escape()同时还会对锚(anchor)转义，即类似"url#anchor"的URL，因此，你不能用 `mod_rewrite` 对此类URL直接重定向。那么如何实现呢?

方案：

必须用NPH-CGI脚本使它自己重定向，因为对NPH(无须解析的HTTP头)不会发生转义操作。首先，在针对服务器的配置中(应该位于所有重写规则的最后)，引入一种新的URL类型"xredirect:"：

```
RewriteRule ^xredirect:(.+) /path/to/nph-xredirect.cgi/$1 \
            [T=application/x-httpd-cgi,L]
```

以强制所有带"xredirect:"前缀的URL被传送到如下的nph-xredirect.cgi程序：

```
#!/path/to/perl
##
##  nph-xredirect.cgi -- NPH/CGI script for extended redirects

##

$| = 1;
$url = $ENV{'PATH_INFO'};

print "HTTP/1.0 302 Moved Temporarily\n";
print "Server: $ENV{'SERVER_SOFTWARE'}\n";
print "Location: $url\n";
print "Content-type: text/html\n";
print "\n";
print "&lt;html&gt;\n";
print "&lt;head&gt;\n";
print "&lt;title&gt;302 Moved Temporarily (EXTENDED)&lt;/title&gt;\n";
print "&lt;/head&gt;\n";
print "&lt;body&gt;\n";
print "&lt;h1&gt;Moved Temporarily (EXTENDED)&lt;/h1&gt;\n";
print "The document has moved &lt;a HREF=\"$url\"&gt;here&lt;/a&gt;.&lt;p&gt;\n";
print "&lt;/body&gt;\n";
print "&lt;/html&gt;\n";

##EOF##
```

这是一种可以重定向所有URL类型的方法，包括不被 `mod_rewrite` 直接支持的类型。所以，还可以这样重定向"news:newsgroup"：

```
RewriteRule ^anyurl  xredirect:news:newsgroup
```

注意：无须对上述规则加[R]或[R,L]，因为"xredirect:"需要在稍后被其特殊的"管道传送"规则扩展。

文档访问的多路复用

说明：

你知道http://www.perl.com/CPAN的CPAN(综合Perl存档网络)？它实现了一个重定向以提供全世界的CPAN镜像中离访问者最近的一个FTP站点，也可以称之为FTP访问多路复用服务。CPAN是通过CGI脚本实现的，那么用 `mod_rewrite` 如何实现呢？

方案：

首先，我们注意到 `mod_rewrite` 从3.0.0版本开始，还可以重写"ftp:"类型。其次，对客户端顶级域名的路径最近的求取可以用 `RewriteMap` 实现。利用链式规则集，并用顶级域名作为查找多路复用地图的键，可以这样做：

```
RewriteEngine on
RewriteMap    multiplex                txt:/path/to/map.cxan
RewriteRule   ^/CxAN/(.*)              %{REMOTE_HOST}::$1                 [C]
RewriteRule   ^.+\.**([a-zA-Z]+)**::(.*)$  ${multiplex:**$1**|ftp.default.dom}$2  [R,L]
```

```
##
##  map.cxan -- Multiplexing Map for CxAN
##

de        ftp://ftp.cxan.de/CxAN/
uk        ftp://ftp.cxan.uk/CxAN/
com       ftp://ftp.cxan.com/CxAN/
 :
##EOF##
```

依赖于时间的重写

说明：

在页面内容按时间不同而变化的场合，比如重定向特定页面，许多网管仍然采用CGI脚本的方法，如何用 `mod_rewrite` 来实现呢？

方案：

有许多类似TIME_xxx的变量可以用在重写条件中，利用"<STRING", "
>STRING"和"=STRING"的类型比较，并加以连接，就可以实现依赖于时间的重写：

```
RewriteEngine on
RewriteCond   %{TIME_HOUR}%{TIME_MIN} &gt;0700
RewriteCond   %{TIME_HOUR}%{TIME_MIN} &lt;1900
RewriteRule   ^foo\.html$             foo.day.html
RewriteRule   ^foo\.html$             foo.night.html
```

此例使URLfoo.html在07:00-19:00时指向foo.day.html，而在其余时间，则指向foo.night.html，对主页是一个不错的功能...

对**YYYY**过渡为**XXXX**的向前兼容

说明：

在转变了大批.html文件为.phtml，使文档.YYYY过渡成为文档.XXXX后，如何保持URL的向前兼容(仍然虚拟地存在)？

方案：

只须按基准文件名重写，并测试带有新的扩展名的文件是否存在，如果存在，则用新的，否则，仍然用原来的。

```
#   backward compatibility ruleset for
#   rewriting document.html to document.phtml
#   when and only when document.phtml exists
#   but no longer document.html
RewriteEngine on
RewriteBase   /~quux/
#   parse out basename, but remember the fact
RewriteRule   ^(.*)\.html$              $1      [C,E=WasHTML:yes]
#   rewrite to document.phtml if exists
RewriteCond   %{REQUEST_FILENAME}.phtml -f
RewriteRule   ^(.*)$ $1.phtml                   [S=1]
#   else reverse the previous basename cutout
RewriteCond   %{ENV:WasHTML}            ^yes$
RewriteRule   ^(.*)$ $1.html
```

内容的处理

新旧URL(内部的)

说明:

假定已经把文件 `bar.html` 改名为 `foo.html` ，需要对老的URL向前兼容，即让用户仍然可以使用老的URL，而感觉不到文件被改名了。

方案:

通过以下规则内部地重写老的URL为新的：

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^**foo**\.html$  **bar**.html
```

新旧URL(外部的)

说明:

仍然假定已经把文件 `bar.html` 改名为 `foo.html` ，需要对老的URL向前兼容，但是要让用户得到文件被改名的暗示，即浏览器的地址栏中显示的是新的URL。

方案:

作一个HTTP的强制重定向以改变浏览器和用户界面上的显示：

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^**foo**\.html$  **bar**.html  [**R**]
```

依赖于浏览器的内容

说明:

至少对重要的顶级页面，有时候有必要提供依赖于浏览器的最佳的内容，即对最新的Netscape提供最大化的版本，对Lynx提供最小化的版本，而对其他的浏览器则提供一个功能一般的版本。

方案:

对此，内容协商无能为力，因为浏览器不提供那种形式的类型，所以只能在HTTP头"User-Agent"上想办法。以下规则集可以完成这个操作：如果HTTP头"User-Agent"以"Mozilla/3"开头，则页面 `foo.html` 被重写为 `foo.NS.html` ，而后重写操作终止；如果是"Lynx"或者版本号为1和2的"Mozilla"，则重写为 `foo.20.html` ；而其他所有的浏览器收到的页面则是 `foo.32.html`

```
RewriteCond %{HTTP_USER_AGENT}  ^**Mozilla/3**.*
RewriteRule ^foo\.html$         foo.**NS**.html          [**L**]

RewriteCond %{HTTP_USER_AGENT}  ^**Lynx/**.*         [OR]
RewriteCond %{HTTP_USER_AGENT}  ^**Mozilla/[12]**.*
RewriteRule ^foo\.html$         foo.**20**.html          [**L**]

RewriteRule ^foo\.html$         foo.**32**.html          [**L**]
```

动态镜像

说明:

假定，需要在我们的名称空间里加入其他远程主机的页面。对FTP服务器，可以用 `mirror` 程序以在本地机器上维持一个对远程数据的最新的拷贝；对web服务器，可以用类似的用于HTTP的 `webcopy` 程序。但这两种技术都有一个主要的缺点：此本地拷贝必须通过这个程序的执行来更新。所以，比较好的方法是，不采用静态镜像，而采用动态镜像，即在有数据请求时自动更新(远程主机上更新的数据)。

方案:

为此，使用代理吞吐(<dfn class="calibre27">Proxy Throughput</dfn>)功能(flag `[P]`)，以映射远程页面甚至整个远程网络区域到我们的名称空间：

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^**hotsheet/**(.*)$  **http://www.tstimpreso.com/hotsheet/**$1  [**P**]
```

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^**usa-news\.html**$   **http://www.quux-corp.com/news/index.html**  [**P*
```

反向动态镜像

说明:

...

方案:

```
RewriteEngine on
RewriteCond   /mirror/of/remotesite/$1           -U
RewriteRule   ^http://www\.remotesite\.com/(.*)$ /mirror/of/remotesite/$1
```

通过Intranet取得丢失的数据

说明:

这是一种在受防火墙保护的(内部)Intranet(`www2.quux-corp.dom`)上保存和维护实际数据，而虚拟地运行企业级(外部)Internet web服务器(`www.quux-corp.dom`)的巧妙的方法。这种方法是外部服务器在空闲时间从内部服务器取得被请求的数据。

方案:

首先，必须确保防火墙对内部服务器的保护，并只允许此外部服务器取得数据。对包过滤(packet-filtering)防火墙，可以如下制定防火墙规则：

```
**ALLOW** Host www.quux-corp.dom Port &gt;1024 --&gt; Host www2.quux-corp.dom Port **80**

**DENY**  Host *                 Port *     --&gt; Host www2.quux-corp.dom Port **80**
```

按你的实际配置，只要对上例稍作调整即可。接着，建立通过代理后台获取丢失数据的 `mod_rewrite` 规则：

```
RewriteRule ^/~([^/]+)/?(.*)          /home/$1/.www/$2
RewriteCond %{REQUEST_FILENAME}       **!-f**
RewriteCond %{REQUEST_FILENAME}       **!-d**
RewriteRule ^/home/([^/]+)/.www/?(.*) http://**www2**.quux-corp.dom/~$1/pub/$2 [**P**]
```

负载的均衡

说明:

如何均衡 `www.foo.com` 的负载到 `www[0-5].foo.com` (一共是6个服务器)?

方案:

这个问题有许多可能的解决方案，在此，我们讨论通称为“基于DNS”的方案，和特殊的使用 `mod_rewrite` 的方案:

1. **DNS循环(DNS Round-Robin)**

   最简单的方法是用 `BIND` 的DNS循环特性，只要按惯例设置 `www[0-9].foo.com` 的DNS的A(地址)记录，如：

   ```
   www0   IN  A       1.2.3.1
   www1   IN  A       1.2.3.2
   www2   IN  A       1.2.3.3
   www3   IN  A       1.2.3.4
   www4   IN  A       1.2.3.5
   www5   IN  A       1.2.3.6
   ```

   然后，增加以下各项:

   ```
   www    IN  CNAME   www0.foo.com.
          IN  CNAME   www1.foo.com.
          IN  CNAME   www2.foo.com.
          IN  CNAME   www3.foo.com.
          IN  CNAME   www4.foo.com.
          IN  CNAME   www5.foo.com.
          IN  CNAME   www6.foo.com.
   ```

   注意，上述看起来似乎是错误的，但事实上，它的确是 `BIND` 中的一个预期的特性，而且也可以这样用。无论如何，现在 `www.foo.com` 已经被解析， `BIND` 可以给出 `www0-www6` ，虽然每次在次序上会有轻微的置换/循环，客户端的请求可以被分散到各个服务器。但这并不是一个优秀的负载均衡方案，因为DNS解析信息可以被网络中其他名称服务器缓冲，而一旦 `www.foo.com` 被解析为 `wwwN.foo.com` ，则其后继请求都将被送往 `www.foo.com` 。但是最终结果是正确的，因为请求的总量的确被分散到各个服务器了

2. **DNS 负载均衡**

   一种成熟的基于DNS的负载均衡方法是使用http://www.stanford.edu/~schemers/docs/lbnamed/lbnamed.html的 `lbnamed` 程序，它是一个Perl5程序，带有若干辅助工具，实现了真正的基于DNS的负载均衡。

3. **代理吞吐循环(Proxy Throughput Round-Robin)**

   这是一个使用 `mod_rewrite` 及其代理吞吐特性的方法。首先，在DNS记录中将 `www0.foo.com` 固定为 `www.foo.com` ，如下：

   ```
   www    IN  CNAME   www0.foo.com.
   ```

   其次，将 `www0.foo.com` 转换为一个专职代理服务器，即由这个机器把所有到来的URL通过内部代理分散到另外5个服务器(`www1-www5`)。为此，必须建立一个规则集，对所有URL调用一个负载均衡脚本 `lb.pl` 。

```
RewriteEngine on
RewriteMap    lb      prg:/path/to/lb.pl
RewriteRule   ^/(.+)$ ${lb:$1}           [P,L]
```

以下是 `lb.pl` ：

```
#!/path/to/perl
##
##  lb.pl -- load balancing script
##

$| = 1;

$name   = "www";     # the hostname base
$first  = 1;         # the first server (not 0 here, because 0 is myself)
$last   = 5;         # the last server in the round-robin
$domain = "foo.dom"; # the domainname

$cnt = 0;
while (&lt;STDIN&gt;) {
    $cnt = (($cnt+1) % ($last+1-$first));
    $server = sprintf("%s%d.%s", $name, $cnt+$first, $domain);
    print "http://$server/$_";
}

##EOF##
```

最后的说明：这样有用吗？ `www0.foo.com` 似乎也会超载呀？答案是：没错，它的确会超载，但是它超载的仅仅是简单的代理吞吐请求！所有诸如SSI、CGI、ePerl等等的处理完全是由其他机器完成的，这个才是要点。 | --- | --- |

4. 硬件/**TCP**循环

   还有一个硬件解决方案。Cisco有一个叫LocalDirector的东西，实现了TCP/IP层的负载均衡，事实上，它是一个位于网站集群前端的电路级网关。如果你有足够资金而且的确需要高性能的解决方案，那么可以用这个。

反向代理

说明:

...

方案:

```
##
##  apache-rproxy.conf -- Apache configuration for Reverse Proxy Usage
##

#   server type
ServerType           standalone
Listen               8000
MinSpareServers      16
StartServers         16
MaxSpareServers      16
MaxClients           16
```

```
MaxRequestsPerChild  100

#   server operation parameters
KeepAlive            on
MaxKeepAliveRequests 100
KeepAliveTimeout     15
Timeout              400
IdentityCheck        off
HostnameLookups      off

#   paths to runtime files
PidFile              /path/to/apache-rproxy.pid
LockFile             /path/to/apache-rproxy.lock
ErrorLog             /path/to/apache-rproxy.elog
CustomLog            /path/to/apache-rproxy.dlog "%{%v/%T}t %h -&gt; %{SERVER}e URL: %U"

#   unused paths
ServerRoot           /tmp
DocumentRoot         /tmp
CacheRoot            /tmp
RewriteLog           /dev/null
TransferLog          /dev/null
TypesConfig          /dev/null
AccessConfig         /dev/null
ResourceConfig       /dev/null

#   speed up and secure processing
&lt;Directory /&gt;
Options -FollowSymLinks -SymLinksIfOwnerMatch
AllowOverride None

&lt;/Directory&gt;

#   the status page for monitoring the reverse proxy
&lt;Location /apache-rproxy-status&gt;
SetHandler server-status
&lt;/Location&gt;

#   enable the URL rewriting engine
RewriteEngine        on
RewriteLogLevel      0

#   define a rewriting map with value-lists where
#   mod_rewrite randomly chooses a particular value
RewriteMap     server  rnd:/path/to/apache-rproxy.conf-servers

#   make sure the status page is handled locally
#   and make sure no one uses our proxy except ourself
RewriteRule    ^/apache-rproxy-status.*  -  [L]
RewriteRule    ^(http|ftp)://.*          -  [F]

#   now choose the possible servers for particular URL types
RewriteRule    ^/(.*\.(cgi|shtml))$  to://${server:dynamic}/$1  [S=1]
RewriteRule    ^/(.*)$               to://${server:static}/$1

#   and delegate the generated URL by passing it
#   through the proxy module
RewriteRule    ^to://([^/]+)/(.*)    http://$1/$2   [E=SERVER:$1,P,L]

#   and make really sure all other stuff is forbidden
#   when it should survive the above rules...
RewriteRule    .*                    -              [F]

#   enable the Proxy module without caching
ProxyRequests        on
NoCache              *

#   setup URL reverse mapping for redirect reponses
ProxyPassReverse  /  http://www1.foo.dom/
ProxyPassReverse  /  http://www2.foo.dom/
ProxyPassReverse  /  http://www3.foo.dom/
ProxyPassReverse  /  http://www4.foo.dom/
```

```
ProxyPassReverse  /  http://www5.foo.dom/
ProxyPassReverse  /  http://www6.foo.dom/
```

```
##
##  apache-rproxy.conf-servers -- Apache/mod_rewrite selection table
##

#   list of backend servers which serve static
#   pages (HTML files and Images, etc.)
static    www1.foo.dom|www2.foo.dom|www3.foo.dom|www4.foo.dom

#   list of backend servers which serve dynamically
#   generated page (CGI programs or mod_perl scripts)
dynamic   www5.foo.dom|www6.foo.dom
```

新的MIME类型，新的服务

说明:

网上有许多很巧妙的CGI程序，但是用法晦涩，许多网管弃之不用。即使是Apache的MEME类型的动作处理器，也仅仅在CGI程序不需要在其输入中包含特殊URL(`PATH_INFO` 和 `QUERY_STRINGS`)时才很好用。首先，配置一种新的后缀为 `.scgi` (安全CGI)文件类型，其处理器是很常见的 `cgiwrap` 程序。问题是：如果使用同类URL规划(见上述)，而用户宿主目录中的一个文件的URL是 `/u/user/foo/bar.scgi` ，可是 `cgiwrap` 要求的URL的格式是 `/~user/foo/bar.scgi/` ，以下规则解决了这个问题：

```
RewriteRule ^/[uge]/**([^/]+)**/\.www/(.+)\.scgi(.*) ...
... /internal/cgi/user/cgiwrap/~**$1**/$2.scgi$3  [NS,**T=application/x-http-cgi**]
```

另外，假设需要使用其他程序： `wwwlog` (显示 `access.log` 中的一个URL子树)和 `wwwidx` (对一个URL子树运行Glimpse)，则必须对这些程序提供URL区域作为其操作对象。比如，对 `/u/user/foo/` 执行 `swwidx` 程序的超链是这样的：

```
/internal/cgi/user/swwidx?i=/u/user/foo/
```

其缺点是，必须同时硬编码超链中的区域和CGI的路径，如果重组了这个区域，就需要花费大量时间来修改各个超链。

方案:

方案是用一个特殊的新的URL格式，自动拼装CGI参数：

```
RewriteRule   ^/([uge])/([^/]+)(/?.*)/\*  /internal/cgi/user/wwwidx?i=/$1/$2$3/
RewriteRule   ^/([uge])/([^/]+)(/?.*):log /internal/cgi/user/wwwlog?f=/$1/$2$3
```

现在，这个搜索到 `/u/user/foo/` 的超链简化成了：

```
HREF="*"
```

它会被内部地自动转换为

```
/internal/cgi/user/wwwidx?i=/u/user/foo/
```

如此，可以为使用" `:log` "的超链，拼装出调用CGI程序的参数。

从静态到动态

说明:

如何无缝转换静态页面 `foo.html` 为动态的 `foo.cgi` ，而不为浏览器/用户所察觉。

方案:

只须重写此URL为CGI-script，以强制为可以作为CGI-script运行的正确的MIME类型。如此，对 `/~quux/foo.html` 的请求其实会执行 `/~quux/foo.cgi` 。

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^foo\.**html**$  foo.**cgi**  [T=**application/x-httpd-cgi**]
```

传输过程中的内容协商

说明:

这是一个很难解的功能：动态生成的静态页面，即它应该作为静态页面发送(从文件系统中读出，然后直接发出去)，但是如果它丢失了，则由服务器动态生成。这样，可以静态地提供CGI生成的页面，除非有人(或者是一个cronjob)删除了这些静态页面，而且其内容可以得到更新。

方案:

以下规则集实现了这个功能：

```
RewriteCond %{REQUEST_FILENAME}   **!-s**
RewriteRule ^page\.**html**$          page.**cgi**   [T=application/x-httpd-cgi,L]
```

这样，如果 `page.html` 不存在或者文件大小为null，则对 `page.html` 的请求会导致 `page.cgi` 的运行。其中奥妙在于 `page.cgi` 是一个将输出写入 `page.html` 的(同时也写入 `STDOUT`)的常规的CGI脚本，执行完毕，服务器则将 `page.html` 的内容发出。如果网管需要强制更新其内容，只须删除 `page.html` 即可(通常由一个cronjob完成)。

自动更新的文档

说明:

建立一个复杂的页面，能够在用编辑器写了一个更新的版本时自动在浏览器上得到刷新，这不是很好吗？这可能吗？

方案:

这是可行的! 这需要综合利用MIME多成分、web服务器的NPH和 `mod_rewrite` 的URL操控特性。首先，建立一个新的URL特性：对在文件系统中更新时需要刷新的所有URL加上" `:refresh` "。

```
RewriteRule   ^(/[uge]/[^/]+/?.*):refresh  /internal/cgi/apache/nph-refresh?f=$1
```

然后，修改URL

```
/u/foo/bar/page.html:refresh
```

以内部地操控此URL

```
/internal/cgi/apache/nph-refresh?f=/u/foo/bar/page.html
```

接着就是NPH-CGI脚本部分了。虽然，人们常说"将此作为一个练习留给读者"，但我还是给出答案了。

```
#!/sw/bin/perl
##
##  nph-refresh -- NPH/CGI script for auto refreshing pages
##  Copyright (c) 1997 Ralf S. Engelschall, All Rights Reserved.
##
$| = 1;

#   split the QUERY_STRING variable
@pairs = split(/&/, $ENV{'QUERY_STRING'});
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $name =~ tr/A-Z/a-z/;
    $name = 'QS_' . $name;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
    eval "\$$name = \"$value\"";
}
$QS_s = 1 if ($QS_s eq '');
$QS_n = 3600 if ($QS_n eq '');
if ($QS_f eq '') {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "&lt;b&gt;ERROR&lt;/b&gt;: No file given\n";
    exit(0);
}
if (! -f $QS_f) {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "&lt;b&gt;ERROR&lt;/b&gt;: File $QS_f not found\n";
    exit(0);
```

```
}

sub print_http_headers_multipart_begin {
    print "HTTP/1.0 200 OK\n";
    $bound = "ThisRandomString12345";
    print "Content-type: multipart/x-mixed-replace;boundary=$bound\n";
    &print_http_headers_multipart_next;
}

sub print_http_headers_multipart_next {
    print "\n--$bound\n";
}

sub print_http_headers_multipart_end {
    print "\n--$bound--\n";
}

sub displayhtml {
    local($buffer) = @_;
    $len = length($buffer);
    print "Content-type: text/html\n";
    print "Content-length: $len\n\n";
    print $buffer;
}

sub readfile {
    local($file) = @_;
    local(*FP, $size, $buffer, $bytes);
    ($x, $x, $x, $x, $x, $x, $x, $size) = stat($file);
    $size = sprintf("%d", $size);
    open(FP, "&lt;$file");
    $bytes = sysread(FP, $buffer, $size);
    close(FP);
    return $buffer;
}

$buffer = &readfile($QS_f);
&print_http_headers_multipart_begin;
&displayhtml($buffer);

sub mystat {
    local($file) = $_[0];
    local($time);

    ($x, $x, $x, $x, $x, $x, $x, $x, $x, $mtime) = stat($file);
    return $mtime;
}

$mtimeL = &mystat($QS_f);
$mtime = $mtime;
for ($n = 0; $n &lt; $QS_n; $n++) {
    while (1) {
        $mtime = &mystat($QS_f);
        if ($mtime ne $mtimeL) {
            $mtimeL = $mtime;
            sleep(2);
            $buffer = &readfile($QS_f);
            &print_http_headers_multipart_next;
            &displayhtml($buffer);
            sleep(5);
            $mtimeL = &mystat($QS_f);
            last;
        }
        sleep($QS_s);
    }
}

&print_http_headers_multipart_end;

exit(0);

##EOF##
```

大型虚拟主机

说明:

Apache的 `<VirtualHost>` 功能很强，在有几十个虚拟主机的情况下运行得很好，但是如果你是ISP，需要提供几百个虚拟主机，那么这就不是一个最佳的选择了。

方案:

为此，需要用<dfn class="calibre27">代理吞吐(Proxy Throughput)</dfn>功能(flag `[P]`)映射远程页面甚至整个远程网络区域到自己的名称空间：

```
##
##  vhost.map
##
www.vhost1.dom:80  /path/to/docroot/vhost1
www.vhost2.dom:80  /path/to/docroot/vhost2
     :
www.vhostN.dom:80  /path/to/docroot/vhostN
```

```
##
##  httpd.conf
##
    :
#   use the canonical hostname on redirects, etc.
UseCanonicalName on

    :
#   add the virtual host in front of the CLF-format
CustomLog  /path/to/access_log  "%{VHOST}e %h %l %u %t \"%r\" %&gt;s %b"
    :

#   enable the rewriting engine in the main server
RewriteEngine on

#   define two maps: one for fixing the URL and one which defines
#   the available virtual hosts with their corresponding
#   DocumentRoot.
RewriteMap    lowercase    int:tolower
RewriteMap    vhost        txt:/path/to/vhost.map

#   Now do the actual virtual host mapping
#   via a huge and complicated single rule:
#
#   1\. make sure we don't map for common locations
RewriteCond   %{REQUEST_URL}  !^/commonurl1/.*
RewriteCond   %{REQUEST_URL}  !^/commonurl2/.*
    :
RewriteCond   %{REQUEST_URL}  !^/commonurlN/.*
#
#   2\. make sure we have a Host header, because
#      currently our approach only supports
#      virtual hosting through this header
RewriteCond   %{HTTP_HOST}  !^$
#
#   3\. lowercase the hostname
RewriteCond   ${lowercase:%{HTTP_HOST}|NONE}  ^(.+)$
#
#   4\. lookup this hostname in vhost.map and
#      remember it only when it is a path
#      (and not "NONE" from above)
RewriteCond   ${vhost:%1}  ^(/.*)$
#
#   5\. finally we can map the URL to its docroot location
#      and remember the virtual host for logging puposes
RewriteRule   ^/(.*)$   %1/$1  [E=VHOST:${lowercase:%{HTTP_HOST}}]
    :
```

对访问的限制

阻止Robots

说明:

如何阻止一个完全匿名的robot取得特定网络区域的页面？一个 `/robots.txt` 文件可以包含若干"robot排除协议"的行，但不足以阻止此类robot。

方案:

可以用一个规则集以拒绝对网络区域 `/~quux/foo/arc/` (对一个很深的目录区域进行列表可能会使服务器产生很大的负载)的访问。还必须确保仅阻止特定的robot，就是说，仅仅阻止robot访问主机是不够的，这样会同时也阻止了用户访问该主机。为此，就需要对HTTP头的User-Agent信息作匹配。

```
RewriteCond %{HTTP_USER_AGENT}   ^**NameOfBadRobot**.*
RewriteCond %{REMOTE_ADDR}       ^**123\.45\.67\.[8-9]**$
RewriteRule ^**/~quux/foo/arc/**.+   -   [**F**]
```

阻止内嵌的图片

说明:

假设， `http://www.quux-corp.de/~quux/` 有一些内嵌图片的页面，这些图片很好，所以就有人用超链连到他们自己的页面中了。由于这样徒然增加了我们的服务器的流量，因此，我们不愿意这种事情发生。

方案:

虽然，我们不能100%地保护这些图片不被写入别人的页面，但至少可以对发出HTTP Referer头的浏览器加以限制。

```
RewriteCond %{HTTP_REFERER} **!^$**
RewriteCond %{HTTP_REFERER} !^http://www.quux-corp.de/~quux/.*$ [NC]
RewriteRule **.*\.gif$**        -                                    [F]
```

```
RewriteCond %{HTTP_REFERER}         !^$
RewriteCond %{HTTP_REFERER}         !.*/foo-with-gif\.html$
RewriteRule **^inlined-in-foo\.gif$**   -                        [F]
```

对主机的拒绝

说明:

如何拒绝一批外部列表中的主机对我们服务器的使用？

方案:

```
RewriteEngine on
RewriteMap    hosts-deny  txt:/path/to/hosts.deny
RewriteCond   ${hosts-deny:%{REMOTE_HOST}|NOT-FOUND} !=NOT-FOUND [OR]
RewriteCond   ${hosts-deny:%{REMOTE_ADDR}|NOT-FOUND} !=NOT-FOUND
RewriteRule   ^/.*  -  [F]
```

对代理的拒绝

说明:

如何拒绝某个主机或者来自特定主机的用户使用Apache代理？

方案:

首先，要确保Apache web服务器在编译时配置文件中 `mod_rewrite` 在 `mod_proxy` 的下面！使它在 `mod_proxy` 之前被调用。然后，如下拒绝某个主机...

```
RewriteCond %{REMOTE_HOST} **^badhost\.mydomain\.com$**
RewriteRule !^http://[^/.]\.mydomain.com.*  - [F]
```

...如下拒绝user@host-dependent:

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST}  **^badguy@badhost\.mydomain\.com$**
RewriteRule !^http://[^/.]\.mydomain.com.*  - [F]
```

特殊的认证

说明:

有时候，会需要一种非常特殊的认证，即对一组明确指定的用户，允许其访问，而没有(在使用 `mod_authz_host` 的基本认证方法时可能会出现的)任何提示。

方案:

可是使用一个重写条件列表来排除所有的朋友：

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} **!^friend1@client1.quux-corp\.com$**
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} **!^friend2**@client2.quux-corp\.com$
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} **!^friend3**@client3.quux-corp\.com$
RewriteRule ^/~quux/only-for-friends/      -                                 [F]
```

基于提交者(**Referer**)的反射器

说明:

如何配置一个基于HTTP头"Referer"的反射器以反射到任意数量的提交页面?

方案:

使用这个很巧妙的规则集...

```
RewriteMap  deflector txt:/path/to/deflector.map

RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}} ^-$
RewriteRule ^.* %{HTTP_REFERER} [R,L]

RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}|NOT-FOUND} !=NOT-FOUND
RewriteRule ^.* ${deflector:%{HTTP_REFERER}} [R,L]
```

... 并结合对应的重写映射地图:

```
##
##  deflector.map
##

http://www.badguys.com/bad/index.html    -
http://www.badguys.com/bad/index2.html   -
http://www.badguys.com/bad/index3.html   http://somewhere.com/
```

它可以自动将请求(在映射地图中指定了" `-` "值的时候)反射回其提交页面，或者(在映射地图中URL有第二个参数时)反射到一个特定的URL。

其他

外部重写引擎

说明:

一个常见的问题是如何解决似乎无法用 `mod_rewrite` 解决的FOO/BAR/QUUX/之类的问题？

方案:

可以使用一个与 `RewriteMap` 功能相同的外部 `RewriteMap` 程序，一旦它在Apache启动时被执行，则从 `STDIN` 接收被请求的URL，并将处理过(通常是重写过的)的URL(以相同顺序)在 `STDOUT` 输出。

```
RewriteEngine on
RewriteMap    quux-map       **prg:**/path/to/map.quux.pl
RewriteRule   ^/~quux/(.*)$  /~quux/**${quux-map:$1}**
```

```
#!/path/to/perl

#   disable buffered I/O which would lead
#   to deadloops for the Apache server
$| = 1;

#   read URLs one per line from stdin and
#   generate substitution URL on stdout
while (&lt;&gt;) {
    s|^foo/|bar/|;
    print $_;
}
```

这是一个作演示的例子，只是把所有的URL `/~quux/foo/...` 重写为 `/~quux/bar/...` ，而事实上，可以把它修改以获得任何你需要的功能。但是要注意，虽然一般用户都可以使用，可是只有系统管理员才可以定义这样的地图。

Apache虚拟主机文档

术语"<cite class="calibre27">虚拟主机</cite>"是指在一个机器上运行多个网站(比如：`www.company1.com` 和 `www.company2.com`)。如果每个网站拥有不同的IP地址，则虚拟主机可以是"基于IP"的；如果只有一个IP地址，也可以是"基于主机名"的，其实现对最终用户是透明的。

Apache是率先支持基于IP的虚拟主机的服务器之一。1.1及其更新版本同时支持基于IP和基于主机名的虚拟主机，今后，不同的虚拟主机有时会被称为"基于主机"或"非*IP*虚拟主机"。

下列文档会阐述Apache1.3及其更新版本所支持的虚拟主机的所有细节。

虚拟主机支持

- 基于主机名的虚拟主机(一个IP地址，多个网站)
- 基于IP地址的虚拟主机(每个站点拥有一个的独立IP地址)
- 虚拟主机的普通配置示例
- 文件描述符限制(在日志文件过多的情况下会产生的限制)
- 动态配置大量虚拟主机
- 深入讨论虚拟主机的匹配

配置指令

- `<VirtualHost>`
- `NameVirtualHost`
- `ServerName`
- `ServerAlias`
- `ServerPath`

如果要调试你的虚拟主机配置，你会发现Apache的 `-S` 命令行开关很有用。比如：

```
/usr/local/apache2/bin/httpd -S
```

此命令会输出Apache解析配置文件的详细描述，仔细检查IP地址和主机名会有助于纠正配置错误。(其他命令行参数详见： `httpd` 文档)

基于主机名的虚拟主机

本文档说明了如何使用基于域名的虚拟主机。

基于域名的虚拟主机和基于IP的虚拟主机比较

基于IP的虚拟主机使用连接的IP地址来决定相应的虚拟主机。这样，你就需要为每个虚拟主机分配一个独立的IP地址。而基于域名的虚拟主机是根据客户端提交的HTTP头中标识主机名的部分决定的。使用这种技术，很多虚拟主机可以共享同一个IP地址。

基于域名的虚拟主机相对比较简单，因为你只需要配置你的DNS服务器将每个主机名映射到正确的IP地址，然后配置Apache HTTP服务器，令其辨识不同的主机名就可以了。基于域名的服务器也可以缓解IP地址不足的问题。所以，如果没有特殊原因使你必须使用基于IP的虚拟主机，您最好还是使用基于域名的虚拟主机。下列情况下，你可能会想要使用基于IP的虚拟主机：

- 一些古董级的客户端与基于域名的虚拟主机不兼容。为了与基于域名的虚拟主机兼容，客户端必须发送"Host"头。HTTP/1.1规范中对此做了要求。而所有现在常见的仅支持HTTP/1.0的旧版本浏览器都以附加的方式实现了这个要求。如果你又想支持这些老浏览器，又想使用基于域名的虚拟主机。我们提供了一个技术方案，你可以在本文末尾看到它。
- SSL协议先天特性决定了基于域名的虚拟主机无法成为SSL安全服务器。
- 一些操作系统和网络设备实现的带宽管理技术无法在多个主机共享一个IP的情况下区别它们。

使用基于域名的虚拟主机

相关模块

- `core`

相关指令

- `DocumentRoot`
- `NameVirtualHost`
- `ServerAlias`
- `ServerName`
- `ServerPath`
- `<VirtualHost>`

为了使用基于域名的虚拟主机，你必须指定服务器IP地址(和可能的端口)来使主机接受请求，这个可以用 `NameVirtualHost` 指令来进行配置。如果服务器上所有的IP地址都会用到，你可以用" `*` "作为 `NameVirtualHost` 的参数。如果你打算使用多端口(如运行SSL)你必须在参数中指定一个端口号，比如" `*:80` "。请注意，在 `NameVirtualHost` 指令中指定IP地址并不会使服务器自动侦听那个IP地址。请参阅设置Apache使用的地址和端口一章获取更多详情。另外，这里设定的IP地址必须对应服务器上的一个网络接口。

下一步就是为每个虚拟主机建立 `<VirtualHost>` 段。 `<VirtualHost>` 的参数与 `NameVirtualHost` 的参数必须是一样的(比如说，一个IP地址或" `*` "代表的所有地址)。在每个 `<VirtualHost>` 段中，至少要有一个 `ServerName` 指令来指定伺服哪个主机和一个 `DocumentRoot` 指令来说明这个主机的内容位于文件系统的什么地方。

取消中心主机(Mainhost)

如果你想在现有的web服务器上增加虚拟主机，你必须也为现存的主机建造一个 `<VirtualHost>` 定义块。这个虚拟主机中 `ServerName` 和 `DocumentRoot` 所包含的内容应该与全局的 `ServerName` 和 `DocumentRoot` 保持一致。还要把这个虚拟主机放在配置文件的最前面，来让它扮演默认主机的角色。

比如说，假设你正在为域名 `www.domain.tld` 提供服务，而你又想在同一个IP地址上增加一个名叫 `www.otherdomain.tld` 的虚拟主机，你只需在 `httpd.conf` 中加入以下内容：

```
        NameVirtualHost *:80

        <VirtualHost *:80>

 ServerName www.domain.tld

            ServerAlias domain.tld *.domain.tld

            DocumentRoot /www/domain
        </VirtualHost>

        <VirtualHost *:80>

ServerName www.otherdomain.tld

            DocumentRoot /www/otherdomain
        </VirtualHost>
```

当然，你可以用一个固定的IP地址来代替 `NameVirtualHost` 和 `<VirtualHost>` 指令中的" `*` "号，以达到一些特定的目的。比如说，你可能会希望在一个IP地址上运行一个基于域名的虚拟主机，而在另外一个IP地址上运行一个基于IP的或是另外一套基于域名的虚拟主机。

很多服务器希望自己能通过不只一个域名被访问。我们可以把 `ServerAlias` 指令放入 `<VirtualHost>` 小节中来解决这个问题。比如说在上面的第一个 `<VirtualHost>` 配置段中 `ServerAlias` 指令中列出的名字就是用户可以用来访问同一个web站点的其它名字：

```
        ServerAlias domain.tld *.domain.tld
```

这样，所有对域 `domain.tld` 的访问请求都将由虚拟主机 `www.domain.tld` 处理。通配符标记" `*` "和" `?` "可以用于域名的匹配。当然你不能仅仅搞个名字然后把它放到 `ServerName` 或 `ServerAlias` 里就算完了。你必须先在你的DNS服务器上进行配置，将这些名字和您服务器上的一个IP地址建立映射关系。

最后，你可以把其他一些指令放入 `<VirtualHost>` 段中，以更好的配置一个虚拟主机。大部分指令都可以放入这些 `<VirtualHost>` 段中以改变相应虚拟主机配置。如果您想了解一个特定的指令是否可以这样运用，请参见指令的作用域。主服务器*(main server)*范围内的配置指令(在所有 `<VirtualHost>` 配置段之外的指令)仅在它们没有被虚拟主机的配置覆盖时才起作用。

这样，当一个请求到达的时候，服务器会首先检查它是否使用了一个能和 `NameVirtualHost` 相匹配的IP地址。如果能够匹配，它就会查找每个与这个IP地址相对应的 `<VirtualHost>` 段，并尝试找出一个与请求的主机名相同的 `ServerName` 或 `ServerAlias` 配置项。如果找到了，它就会使用这个服务器。否则，将使用符合这个IP地址的第一个列出的虚拟主机。

综上所述，第一个列出的虚拟主机充当了默认虚拟主机的角色。当一个IP地址与 `NameVirtualHost` 指令中的配置相符的时候，主服务器中的 `DocumentRoot` 将永远不会被用到。所以，如果你想创建一段特殊的配置用于处理不对应任何一个虚拟主机的请求的话，你只要简单的把这段配置放到 `<VirtualHost>` 段中，并把它放到配置文件的最前面就可以了。

与旧版浏览器的兼容性

前面提过，有些浏览器无法对基于域名的虚拟主机发送必要的数据，从而使其无法正常工作。这些浏览器将会收到由配置中符合那个IP地址的第一个列出的虚拟主机发出的页面(基于域名的<cite class="calibre27">主虚拟主机</cite>)。

究竟什么算旧？

请注意，当我们说到旧的时候，我们并不是真的说它们很古老。其实现实中您未必就能用上这些浏览器。现在几乎所有的浏览器都会发送基于域名的虚拟主机所必须的 `Host` 头了。

虽然有点麻烦。但您还是有可能会用到 `ServerPath` 指令，以下是一个配置实例：

```
        NameVirtualHost 111.22.33.44

        <VirtualHost 111.22.33.44>

ServerName www.domain.tld

            ServerPath /domain

            DocumentRoot /web/domain
        </VirtualHost>
```

以上这些说明了什么呢？它说明一个具有" `/domain` "开头的任何URI都会为 `www.domain.tld` 这个虚拟主机所伺服。这意味着这个页面可以由 `http://www.domain.tld/domain/` 的形式为所有的浏览器所访问。能够发送" `Host:` "头的浏览器也能使用 `http://www.domain.tld/` 这种形式来访问它。

为了达到这样的目的。您先要在您的主虚拟主机的页面上放一个到 `http://www.domain.tld/domain/` 的链接。然后，确保在虚拟主机的页面中使用的全是相对链接(诸如：" `file.html` "或" `../icons/image.gif` ")或者是包含 `/domain/` 这个前缀(比如：" `http://www.domain.tld/domain/misc/file.html` "或" `/domain/misc/file.html` ")。

完成这些可能需要一些尝试，但遵照上述指导将会确保你的页面能够为所有的浏览器所正确显示，不论新旧。

基于IP地址的虚拟主机

系统需求

就像它的名字"<cite class="calibre27">基于IP</cite>"所暗示的那样，这样的服务器中每个基于IP的虚拟主机必须拥有不同的IP地址。可以通过配备多个真实的物理网络接口来达到这一要求，也可以使用几乎所有流行的操作系统都支持的虚拟界面来达到这一要求(详情请参见您的系统文档，这种功能一般被称作"IP别名"，一般用"ifconfig"命令来进行设置)。

如何配置Apache

有两种配置方法来使apache支持多主机：为每个虚拟主机运行不同的 `httpd` 守护进程；或者用同一个守护进程来支持所有虚拟主机。

以下情况使用多个守护进程：

- 出于安全的考虑，比如说公司甲不希望公司乙的任何人能用除web以外的方式访问到他们的数据。在这种情况下，您需要启动两个守护进程。每个进程都使用不同的 `User` ， `Group` ， `Listen` ， `ServerRoot` 设置。
- 您能够为机器上的每个IP地址提供内存和文件描述符需求。您只能 `Listen` 一个"通配符型"地址或一个特定的地址。所以不管出于什么原因，如果您需要侦听一个特定的地址，您就必须同时侦听所有特定的地址。(尽管可以让一个 `httpd` 侦听N-1个地址，而让另一个侦听剩下的地址)

以下情况使用单一守护进程：

- httpd的配置可以为多个虚拟主机共享而不引起麻烦。
- 机器要接受大量的访问请求，从而多启动一个守护进程会导致性能大幅度降低。

设置多个守护进程

为每个虚拟主机创建一个不同的 `httpd` 安装。每次安装都在配置文件中使用 `Listen` 指令指定守护进程伺服的IP地址(或虚拟主机)。比如：

```
Listen www.smallco.com:80
```

建议您使用IP地址来取代域名(理由请参见关于DNS和Apache)。

配置拥有多个虚拟主机的单一守护进程

在这种情况下，单一的 `httpd` 将伺服所有对主服务器和虚拟主机的请求。而配置文件中的 `VirtualHost` 指令将为每个虚拟主机配置不同的 `ServerAdmin` , `ServerName` , `DocumentRoot` , `ErrorLog` , `TransferLog` , `CustomLog` 。例如：

```
<VirtualHost www.smallco.com>
ServerAdmin webmaster@mail.smallco.com
DocumentRoot /groups/smallco/www
ServerName www.smallco.com
ErrorLog /groups/smallco/logs/error_log
TransferLog /groups/smallco/logs/access_log
</VirtualHost>
<VirtualHost www.baygroup.org>
ServerAdmin webmaster@mail.baygroup.org
DocumentRoot /groups/baygroup/www
ServerName www.baygroup.org
ErrorLog /groups/baygroup/logs/error_log
TransferLog /groups/baygroup/logs/access_log
</VirtualHost>
```

建议您使用IP地址来取代域名(理由请参见关于DNS和Apache)。

除了创建进程的指令和其他一些指令外，几乎所有的配置指令都能用于 `<VirtualHost>` 指令中。您可以使用指令索引在作用域中查询一个指令是否可以用于 `<VirtualHost>` 指令。

如果使用了suEXEC包装，那么 `SuexecUserGroup` 指令也可以在 `<VirtualHost>` 段中使用。

安全警示：当指定日志文件时，请记住有安全风险。一些别有用心的人会在那个目录拥有写权限。请参见安全方面的提示获取详情。

大批量虚拟主机的动态配置

本文档描述如何使用Apache有效的架设大批量虚拟主机。

动机

如果你的配置文件 `httpd.conf` 中包含类似下面的许多 `<VirtualHost>` 段，并且其中的内容都大致相同的话，你应该会对这里所讲的技术感兴趣。比如：

```
NameVirtualHost 111.22.33.44

<VirtualHost 111.22.33.44>

 ServerName                 www.customer-1.com

    DocumentRoot        /www/hosts/www.customer-1.com/docs

    ScriptAlias  /cgi-bin/  /www/hosts/www.customer-1.com/cgi-bin
</VirtualHost>

<VirtualHost 111.22.33.44>

 ServerName                 www.customer-2.com

    DocumentRoot        /www/hosts/www.customer-2.com/docs

    ScriptAlias  /cgi-bin/  /www/hosts/www.customer-2.com/cgi-bin
</VirtualHost>

# 等 等 等 。。。

<VirtualHost 111.22.33.44>

 ServerName                 www.customer-N.com

    DocumentRoot        /www/hosts/www.customer-N.com/docs

    ScriptAlias  /cgi-bin/  /www/hosts/www.customer-N.com/cgi-bin
</VirtualHost>
```

最基本的思想是用动态的机制来实现所有这些静态的 `<VirtualHost>` 配置段。这样做有许多优点：

1. 配置文件变小，使得Apache可以更快的启动，同时消耗更少的内存。
2. 添加一个虚拟主机，应该只是简单的在文件系统中创建合适的目录，以及配置相关的DNS信息，且无需重新启动Apache 。

主要的缺点是你无法针对每个虚拟主机使用不同的日志文件。然而，如果真的在配置有大量虚拟主机的服务器上记录不同的日志文件的话，很有可能会达到操作系统所允许的最大文件描述符的数量。更好的办法是把日志写到管道或者先入先出的栈，并启用其他的进程来分拣所得到的日志信息(同时也可以做一些历史纪录的统计等等)。

概述

一个虚拟主机由两部分来定义：一个是它的IP地址，还有一个是HTTP的" `Host:` "请求头。动态大量虚拟主机的技术，是基于自动在所要返回的文件路径中插入相关信息的想法实现的。使用 `mod_vhost_alias` 可以很容易的实现，但如果你的Apache版本低于1.3.6，则你必须使用 `mod_rewrite` 。两者在默认情况下都不启用；要使用他们，必须在配置和编译Apache的阶段启用。

我们需要做很多"伪装"，才能使动态虚拟主机看起来像普通主机。最重要的一点是Apache使用虚拟主机名(ServerName)来生成自引用(self-referential)URL等信息。这是用 `ServerName` 指令来配置的，并且可以通过环境变量 `SERVER_NAME` 传递给CGI脚本。运行时实际使用的值是由 `UseCanonicalName` 指令的设置来控制的。当 `UseCanonicalName Off` 时，虚拟主机名(ServerName)取自请求中的" `Host:` "头。当 `UseCanonicalName DNS` 时，则通过DNS反解析虚拟主机的IP地址得到主机名。以前的做法是基于名称的动态虚拟主机，现在常用基于IP地址的虚拟主机。如果Apache无法判断虚拟主机名，则可能是没有" `Host:` "头或是DNS解析失败，这样种情况下，Apache将使用配置 `ServerName` 时所填写的主机名。

另一件需要"伪装"的事情是文档根目录(由 `DocumentRoot` 配置并可以通过 `DOCUMENT_ROOT` 环境变量为CGI脚本所使用)。在通常的配置方式下，这些设置信息由核心(core)模块在将URI映射到文件系统的时候使用，但是如果使用动态虚拟主机配置，这些信息将由另外一个使用不同于核心(core)模块将URI映射到文件系统的方式的模块(`mod_vhost_alias` 或 `mod_rewrite`)使用。这两个模块都不负责设置 `DOCUMENT_ROOT` 环境变量，所以如果CGI或SSI程序使用了 `DOCUMENT_ROOT` 环境变量，那么将得到错误的值。

简单的动态虚拟主机

这是 `httpd.conf` 文件中，完成和上文动机部分所提到的虚拟主机一样效果的配置方法，但这里采用了 `mod_vhost_alias` 模块：

```
# 从"Host:"头中取得主机名

UseCanonicalName Off

# 这种日志格式可以从第一个字段中提取出主机名

LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon

CustomLog logs/access_log vcommon

# 在返回请求的文件名路径中包含主机名

VirtualDocumentRoot /www/hosts/%0/docs

VirtualScriptAlias  /www/hosts/%0/cgi-bin
```

将 `UseCanonicalName Off` 的配置改为 `UseCanonicalName DNS` 即可实现基于IP地址的虚拟主机。而在文件路径中所要插入的服务器名则通过虚拟主机的IP地址解析得到。

一个实际的个人主页系统

这里对上面的系统作了一点调整，便可作为ISP的个人主页服务器。我们使用了略微复杂的方法，从主机名(ServerName)中提取子字符串，并插入到文件路径中。在这个例子中 `www.user.isp.com` 的文档将在 `/home/user/` 中定位。并对所有虚拟主机使用单个 `cgi-bin` 目录。

```
# 所有之前的准备事项和上面一样，然后在文件路径中包含主机名

VirtualDocumentRoot /www/hosts/%2/docs

# 单个cgi-bin目录

ScriptAlias  /cgi-bin/  /www/std-cgi/
```

更复杂的关于 `VirtualDocumentRoot` 的设置，可以查阅 `mod_vhost_alias` 文档。

在同一个服务器上架设多个主机的虚拟系统

更复杂的设置，应该使用Apache的 `<VirtualHost>` 容器来管理各种虚拟主机配置的作用域。例如，你可以用一个IP地址来给个人主页客户使用，同时用下面的配置提供给商业客户使用。自然的，这两者通过运用 `<VirtualHost>` 结合到一起。

```
UseCanonicalName Off

LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon

<Directory /www/commercial>

 Options FollowSymLinks

    AllowOverride All
</Directory>

<Directory /www/homepages>

 Options FollowSymLinks

    AllowOverride None
</Directory>

<VirtualHost 111.22.33.44>

 ServerName www.commercial.isp.com

    CustomLog logs/access_log.commercial vcommon

    VirtualDocumentRoot /www/commercial/%0/docs

    VirtualScriptAlias  /www/commercial/%0/cgi-bin
</VirtualHost>

<VirtualHost 111.22.33.45>

 ServerName www.homepages.isp.com

    CustomLog logs/access_log.homepages vcommon

    VirtualDocumentRoot /www/homepages/%0/docs

    ScriptAlias         /cgi-bin/ /www/std-cgi/
</VirtualHost>
```

更为有效的基于IP地址的虚拟主机

在第一个例子中说过，转为基于IP地址的虚拟主机设置很容易做到。但不幸的是，那种做法并不高效，因为这样会在每次处理请求时，需要查询DNS。通过在文件系统中包含IP地址的做法可以避免这样的问题。这样一来，免去了和主机名的关联，在日志记录中也一样可以用IP来分离不同日志。Apache将不会为了确定主机名(ServerName)而去做DNS查询。

```
# 从IP地址反解析得到主机名

UseCanonicalName DNS

# 在日志中包含IP地址，便于以后分拣

LogFormat "%A %h %l %u %t \"%r\" %s %b" vcommon

CustomLog logs/access_log vcommon

# 在文件路径中包含IP地址

VirtualDocumentRootIP /www/hosts/%0/docs

VirtualScriptAliasIP  /www/hosts/%0/cgi-bin
```

使用老版本的Apache

上面的例子基于 `mod_vhost_alias` ，但它是在版本1.3.6之后才出现的。如果你的版本比较老，可以通过使用 `mod_rewrite` 来达到相同的目的，如下所示。但只能是基于"Host:"头方式的虚拟主机。

此外还须注意日志方面的问题。Apache1.3.6是第一个支持" `%V` "日志格式指令的版本，在版本1.3.0-1.3.3中，" `%v` "选项做和" `%V` "一样的事情；而在版本1.3.4中没有等价指令。在所有的这些版本中，指令 `UseCanonicalName` 可以出现在 `.htaccess` 文件中，这意味着客户的设置可能会导致日志记录紊乱。所以最好的做法是使用" `%{Host}i` "指令，它可以直接记录" `Host:` "头；注意，这样可能在末尾包含" `:port` "，而使用" `%V` "则不会这样。

使用 `mod_rewrite` 实现简单的动态虚拟主机

这里的例子摘自 `httpd.conf` ，效果等同于第一个例子中的情况。前半部分和上面的例子大致相似，只是为了向后兼容 `mod_rewrite` 作了适当修改；后半部分配置 `mod_rewrite` 来做实际的工作。

有些特别的地方需要注意：默认情况下， `mod_rewrite` 在所有其他URI转换模块(`mod_alias` 等)之前运行，所以如果使用这些模块的话， `mod_rewrite` 必须作相应的调整。同时，我们还要为每个动态虚拟主机变些戏法，使之等效于 `ScriptAlias`

```
# 从"Host:"头获取主机名

UseCanonicalName Off

# 可分拣的日志

LogFormat "%{Host}i %h %l %u %t \"%r\" %s %b" vcommon

CustomLog logs/access_log vcommon

<Directory /www/hosts>

 # 这里需要ExecCGI ，因为我们不能强制CGI以与ScriptAlias相同的方式执行

    Options FollowSymLinks ExecCGI
</Directory>

# 接下来是关键部分

RewriteEngine On

# 来自"Host:"头的ServerName ，可能大小写混杂

RewriteMap  lowercase  int:tolower

## 首先处理普通文档

# 允许变名/icons/起作用，其他变名类同

RewriteCond  %{REQUEST_URI}  !^/icons/

# 允许CGI

RewriteCond  %{REQUEST_URI}  !^/cgi-bin/

# 开始"变戏法"

RewriteRule  ^/(.*)$  /www/hosts/${lowercase:%{SERVER_NAME}}/docs/$1

## 现在处理CGI(我们需要强制使用一个MIME类型)

RewriteCond  %{REQUEST_URI}  ^/cgi-bin/

RewriteRule  ^/(.*)$  /www/hosts/${lowercase:%{SERVER_NAME}}/cgi-bin/$1  [T=application/x

# ok 了！
```

使用 `mod_rewrite` 的个人主页系统

这里的配置完成和第二个例子相同的工作。

```
RewriteEngine on

RewriteMap   lowercase  int:tolower

# 允许CGI工作

RewriteCond  %{REQUEST_URI}  !^/cgi-bin/

# 检查hostname正确与否，之后才能使RewriteRule起作用

RewriteCond  ${lowercase:%{SERVER_NAME}}  ^www\.[a-z-]+\.isp\.com$

# 将虚拟主机名字连接到URI的开头

# [C]表明本次重写的结果将在下一个rewrite规则中使用

RewriteRule  ^(.+)  ${lowercase:%{SERVER_NAME}}$1  [C]

# 现在创建实际的文件名

RewriteRule  ^www\.([a-z-]+)\.isp\.com/(.*) /home/$1/$2

# 定义全局CGI目录

ScriptAlias  /cgi-bin/  /www/std-cgi/
```

使用独立的虚拟主机配置文件

这样的布局利用了 `mod_rewrite` 的高级特性，在独立的虚拟主机配置文件中转换。如此可以更为灵活，但需要较为复杂的设置。

`vhost.map` 文件包含了类似下面的内容：

```
www.customer-1.com  /www/customers/1

www.customer-2.com  /www/customers/2

# ...

www.customer-N.com  /www/customers/N
```

`http.conf` 包含了：

```
RewriteEngine on

RewriteMap   lowercase  int:tolower

# 定义映射文件

RewriteMap   vhost      txt:/www/conf/vhost.map

# 和上面的例子一样，处理别名

RewriteCond  %{REQUEST_URI}               !^/icons/

RewriteCond  %{REQUEST_URI}               !^/cgi-bin/

RewriteCond  ${lowercase:%{SERVER_NAME}}  ^(.+)$

# 这里做基于文件的重新映射

RewriteCond  ${vhost:%1}                  ^(/.*)$

RewriteRule  ^/(.*)$                      %1/docs/$1

RewriteCond  %{REQUEST_URI}               ^/cgi-bin/

RewriteCond  ${lowercase:%{SERVER_NAME}}  ^(.+)$

RewriteCond  ${vhost:%1}                  ^(/.*)$

RewriteRule  ^/(.*)$                      %1/cgi-bin/$1
```

虚拟主机示例

本文档试图解释一些在设置虚拟主机时经常问及的问题。这些示例向你展示了如何在一个服务器上通过基于域名的或是基于IP的虚拟主机来部署多个web站点。另一份关于如何在一个代理服务器后构建基于多个服务器的站点的说明文档也很快就会出来。

在一个IP地址上运行多个基于域名的web站点

您的服务器有只一个IP地址，而在DNS中有很多域名(CNAMES)映射到这个机器。您而您想要在这个机器上运行 `www.example.com` 和 `www.example.org` 两个站点。

注意

在您的Apache服务器配置中创建一个虚拟主机并不会自动在您的DNS中对主机名做相应更新。您必须自己在DNS中添加域名来指向您的IP地址。否则别人是无法看到您的web站点的。您可以在您的 `hosts` 文件中添加这一条目来进行测试，但这种方法仅适用于那些有这些 `hosts` 文件的机器来使用。

服务器配置

```
    # 确保Apache在监听80端口

    Listen 80

    # 为虚拟主机在所有IP地址上监听

    NameVirtualHost *:80

    <VirtualHost *:80>

 DocumentRoot /www/example1

      ServerName www.example.com

      # 你可以在这里添加其他指令
    </VirtualHost>

    <VirtualHost *:80>

 DocumentRoot /www/example2

      ServerName www.example.org

      # 你可以在这里添加其他指令
    </VirtualHost>
```

因为星号匹配所有IP地址，所以主服务器不接收任何请求。因为 `www.example.com` 首先出现在配置文件中，所以它拥有最高优先级，可以认为是<cite class="calibre27">默认</cite>或<cite class="calibre27">主</cite>服务器。这意味着如果一个请求不能与某个 `ServerName` 指令相匹配，它将会由第一个 `<VirtualHost>` 段所伺服。

注意

如果您愿意，您可以用确定的IP地址来取代" `*` "。在这种情况下， `VirtualHost` 的参数必须与 `NameVirtualHost` 的参数相符：

```
NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>

    # 其他 ...
```

然而，当您的IP地址无法确定的时候，使用" `*` "是很方便的，比如说，您的ISP给您配置的是动态IP地址，而您又使用了某种动态域名解析系统时。因为" `*` "匹配任何IP地址，所以在这种情况下，不论IP地址如何变化，您都不需要另外进行配置。

上述配置就是您在绝大多数情况下使用基于域名的虚拟主机时将要用到的。事实上，仅在一种情况下这样的配置不会让您满意：您想为不同的IP地址或是端口提供不同的内容。

在多于一个IP的情况下使用基于域名的虚拟主机。

注意

在这里讨论的任何技术都可以推广到使用任意数量的IP地址。

服务器有两个IP地址。一个(`172.20.30.40`)用于主服务器 `server.domain.com` ， 另外一个(`172.20.30.50`)用于构建两个或多个虚拟主机。

服务器配置

```
    Listen 80
    # "主"服务器运行于：172.20.30.40
    ServerName server.domain.com
    DocumentRoot /www/mainserver
    # 这是另外一个IP地址
    NameVirtualHost 172.20.30.50
    <VirtualHost 172.20.30.50>
DocumentRoot /www/example1
        ServerName www.example.com
        # 你可以在这里添加其他指令 ...
    </VirtualHost>
    <VirtualHost 172.20.30.50>
DocumentRoot /www/example2
        ServerName www.example.org
        # 你可以在这里添加其他指令 ...
    </VirtualHost>
```

任何不是针对 `172.20.30.50` 的请求都将由主服务器来伺服。而提交给 `172.20.30.50` 却没有主机名或没有" `Host:` "头的请求，都将由 `www.example.com` 伺服。

在不同的IP的地址(比如一个内部和一个外部地址)上提供相同的内容

服务器有两个IP地址(`192.168.1.1` 和 `172.20.30.40`)。这个机器位于内部(局域网)网络和外部(广域网)之间。在外部，域名 `server.example.com` 指向外部地址(`172.20.30.40`)，而在内部则指向内部地址(`192.168.1.1`)。

服务器可以为来自内部和外部的请求提供同样的内容，您只需要一个 `<VirtualHost>` 配置段就可以了。

服务器配置

```
    NameVirtualHost 192.168.1.1

    NameVirtualHost 172.20.30.40

    <VirtualHost 192.168.1.1 172.20.30.40>

 DocumentRoot /www/server1

        ServerName server.example.com

        ServerAlias server
    </VirtualHost>
```

现在，从不同的网络提交的请求都会由同一个 `<VirtualHost>` 段来伺服。

注意

在内网中，您可以使用 `server` 这个名字来代替 `server.example.com` 这个全名。

跟上面一样，在上述的例子里，您可以用" `*` "来代替具体的IP地址，这样就可以对所有的地址都返回相同的内容了。

在不同的端口上运行不同的站点

如果您想让同一个IP的不同端口伺服多个域名。您可以借助在 `NameVirtualHost` 指令中定义端口的方法来达到这个目的。如果您想使用不带" `name:port` "的 `<VirtualHost name:port>` 或是直接用 `Listen` 指令，您的配置将无法生效。

服务器配置

```
    Listen 80

    Listen 8080

    NameVirtualHost 172.20.30.40:80

    NameVirtualHost 172.20.30.40:8080

    <VirtualHost 172.20.30.40:80>

ServerName www.example.com

        DocumentRoot /www/domain-80
    </VirtualHost>

    <VirtualHost 172.20.30.40:8080>

ServerName www.example.com

        DocumentRoot /www/domain-8080
    </VirtualHost>

    <VirtualHost 172.20.30.40:80>

ServerName www.example.org

        DocumentRoot /www/otherdomain-80
    </VirtualHost>

    <VirtualHost 172.20.30.40:8080>

ServerName www.example.org

        DocumentRoot /www/otherdomain-8080
    </VirtualHost>
```

建立基于IP的虚拟主机

一个有两个IP地址(`172.20.30.40` 和 `172.20.30.50`)分别对应域名 `www.example.com` 和 `www.example.org` 的配置如下：

服务器配置

```
    Listen 80

    <VirtualHost 172.20.30.40>

DocumentRoot /www/example1

        ServerName www.example.com
    </VirtualHost>

    <VirtualHost 172.20.30.50>

DocumentRoot /www/example2

        ServerName www.example.org
    </VirtualHost>
```

如果存在主服务器，那么对没有出现在任一个 `<VirtualHost>` 段中的请求(比如，对 `localhost` 的请求)都会由主服务器来伺服。

混用基于端口和基于IP的虚拟主机

如果您的服务器有两个IP地址(`172.20.30.40` 和 `172.20.30.50`)分别对应域名 `www.example.com` 和 `www.example.org` 。对每个域名，您都希望在80端口和8080端口发布您的网站。您可以这样配置：

服务器配置

```
    Listen 172.20.30.40:80

    Listen 172.20.30.40:8080

    Listen 172.20.30.50:80

    Listen 172.20.30.50:8080

    <VirtualHost 172.20.30.40:80>

 DocumentRoot /www/example1-80

        ServerName www.example.com
    </VirtualHost>

    <VirtualHost 172.20.30.40:8080>

 DocumentRoot /www/example1-8080

        ServerName www.example.com
    </VirtualHost>

    <VirtualHost 172.20.30.50:80>

 DocumentRoot /www/example2-80

        ServerName www.example.org
    </VirtualHost>

    <VirtualHost 172.20.30.50:8080>

 DocumentRoot /www/example2-8080

        ServerName www.example.org
    </VirtualHost>
```

混用基于域名和基于IP的虚拟主机

您想在一些地址上配置基于域名的虚拟主机而在另外一些地址上配置基于IP的虚拟主机。

服务器配置

```
Listen 80

NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>

DocumentRoot /www/example1

ServerName www.example.com
</VirtualHost>

<VirtualHost 172.20.30.40>

DocumentRoot /www/example2

ServerName www.example.org
</VirtualHost>

<VirtualHost 172.20.30.40>

DocumentRoot /www/example3

ServerName www.example3.net
</VirtualHost>

# IP-based

<VirtualHost 172.20.30.50>

DocumentRoot /www/example4

ServerName www.example4.edu
</VirtualHost>

<VirtualHost 172.20.30.60>

DocumentRoot /www/example5

ServerName www.example5.gov
</VirtualHost>
```

将 `<Virtual_host>` 和 `mod_proxy` 模块一起使用

下面的例子允许一个前端机器代理一个运行在其他机器上的虚拟主机。在如下示例中，在 `192.168.111.2` 机器上配置了一个同名的虚拟主机。这样，万一在同一台机器上代理了多个主机名， `[ProxyPreserveHost](#calibre_link-670) On` 指令能确保指定的主机名顺利通过代理。

```
<VirtualHost *:*>

    ProxyPreserveHost On

    ProxyPass / http://192.168.111.2

    ProxyPassReverse / http://192.168.111.2/

    ServerName hostname.example.com

</VirtualHost>
```

使用" `_default_` "虚拟主机

为所有端口配置" `_default_` "虚拟主机

这样配置可以捕获所有指向没指定的IP地址和端口的请求。比如：一个没被任何虚拟主机使用的地址/端口对。

服务器配置

```
    <VirtualHost _default_:*>

DocumentRoot /www/default
    </VirtualHost>
```

这样一个使用通配符端口的默认虚拟主机可以有效的防止请求被主服务器接收。

如果一个地址/端口对已经被一个基于域名的虚拟主机使用，那么" `_default_` "虚拟主机决不会处理发向这个地址/端口对的请求。如果一个" `Host:` "请求头中包含未知信息，或者干脆就没有，那么它会被第一个基于域名的虚拟主机(也就是在配置文件中最先出现的使用了那个地址/端口对的虚拟主机)处理。

您可以用 `AliasMatch` 或 `RewriteRule` 来重写任何请求，使它指向一个简单信息页面(或脚本)。

为不同的端口配置" `_default_` "虚拟主机

与第一种一样，但我们想让服务器侦听很多端口而第二个" `_default_` "虚拟主机单独侦听80端口。

服务器配置

```
    <VirtualHost _default_:80>

DocumentRoot /www/default80

        # ...
    </VirtualHost>

    <VirtualHost _default_:*>

DocumentRoot /www/default

        # ...
    </VirtualHost>
```

侦听80端口的" `_default_` "虚拟主机(必须出现在所有使用通配符端口的虚拟主机之前)会捕获所有发向一个未指定的IP地址的请求。主服务器将不会用于伺服任何请求。

为单独一个端口配置"`_default_`"虚拟主机

如果我们只想在80端口上建立唯一的一个"`_default_`"虚拟主机，我们应该这样配置：

服务器配置

```
<VirtualHost _default_:80>
DocumentRoot /www/default
...
</VirtualHost>
```

发向一个未指定地址的80端口的请求将会由这个虚拟主机伺服；而发向未设定地址的其他端口的请求则由主服务器伺服。

将一个基于域名的虚拟主机移植为一个基于IP的虚拟主机

如果一个具有 `www.example.org` 域名的虚拟主机(就是基于域名配置示例中的第二个)得到了自己的IP地址。为了避免一些域名服务器或代理服务器在移植期间仍对这个域名做老的解析，我们可以采用一种过渡方法：同时提供新旧两个IP地址的解析。

达到这个目的很简单。因为我们只要简单的把新地址(`172.20.30.50`)加入 `VirtualHost` 指令就行了。

服务器配置

```
    Listen 80

    ServerName www.example.com

    DocumentRoot /www/example1

    NameVirtualHost 172.20.30.40

    <VirtualHost 172.20.30.40 172.20.30.50>

 DocumentRoot /www/example2

        ServerName www.example.org

        # ...
    </VirtualHost>

    <VirtualHost 172.20.30.40>

 DocumentRoot /www/example3

        ServerName www.example.net

        ServerAlias *.example.net

        # ...
    </VirtualHost>
```

现在这个虚拟主机就可以用新地址(表现为一个基于IP的虚拟主机)和旧地址(表现为一个基于域名的虚拟主机)同时进行访问了。

使用 `ServerPath` 指令

如果我们在同一个服务器上运行了两个基于域名的虚拟主机。为了匹配正确的虚拟主机，客户端必须发送正确的" `Host:` "头。而旧的使用HTTP/1.0的客户端无法发送这样的头，这样Apache就无法辨别客户端想要连接哪个虚拟主机(会用主虚拟主机来伺服这个请求)。为了尽量提供向下兼容性，我们可以提供一个主虚拟主机来返回一个页面，在页面中加入指向基于域名的虚拟主机的URL前缀的链接。

服务器配置

```
    NameVirtualHost 172.20.30.40

    <VirtualHost 172.20.30.40>

 # 主虚拟主机

        DocumentRoot /www/subdomain

        RewriteEngine On

        RewriteRule ^/.* /www/subdomain/index.html

        # ...
    </VirtualHost>

    <VirtualHost 172.20.30.40>

    DocumentRoot /www/subdomain/sub1

 ServerName www.sub1.domain.tld

        ServerPath /sub1/

        RewriteEngine On

        RewriteRule ^(/sub1/.*) /www/subdomain$1

        # ...
    </VirtualHost>

    <VirtualHost 172.20.30.40>

 DocumentRoot /www/subdomain/sub2

        ServerName www.sub2.domain.tld

        ServerPath /sub2/

        RewriteEngine On

        RewriteRule ^(/sub2/.*) /www/subdomain$1

        # ...
    </VirtualHost>
```

由于 `ServerPath` 指令的作用，发送到 `http://www.sub1.domain.tld/sub1/` 的请求总会被sub1-vhost所伺服。 如果客户端发送了正确的" `Host:` "头，发送到 `http://www.sub1.domain.tld/` 的请求只会被sub1-vhost所伺服。如果没有发送" `Host:` "头，客户端将会得到从主虚拟主机发送的信息页面。

请注意，这里还有一点小问题：如果客户端没有发送" `Host:` "头，发送到 `http://www.sub2.domain.tld/sub1/` 的请求还是会被sub1-vhost所伺服。

`RewriteRule` 指令用以确保正确发送了" `Host:` "头的客户端可以任意使用这两种URL变量，比如说：使用或不使用URL前缀。

深入研究虚拟主机的匹配

虚拟主机部分的代码在**Apache 1.3**中进行了完全的重写。本文档试图详细解释Apache在接受到请求后如何确定使用哪一个虚拟主机进行伺服。在新的 `NameVirtualHost` 指令的帮助下，虚拟主机的配置比1.3版以前更加简单和安全。

如果您只是想<cite class="calibre27">让它能够工作</cite>而不愿意进行深入理解，这里有一些示例。

解析配置文件

在 `<VirtualHost>` 配置段外有一个主服务器*(main_server)*段中包含着所有定义。其中有 `<VirtualHost>` 配置段中定义的叫做虚拟主机*(vhost)*的虚拟服务器。

`Listen` , `ServerName` , `ServerPath` , `ServerAlias` 指令可以出现在一个服务器定义段的任何地方。而且每个指令都会覆盖前面出现的同样定义(在那个服务器配置中)。

主服务器段中 `Listen` 指令的默认值是80。主服务器段没有默认的 `ServerPath` 和 `ServerAlias` 指令值。 `ServerName` 的默认值是由服务器的IP地址推断而来。

主服务器的 `Listen` 指令有两个功能：其一是决定Apache将要绑定的网络端口；其二是在重定向中指定绝对URI将使用的端口号。

不象在主服务器里，虚拟服务器的端口不会影响到Apache的监听端口。

每个 `VirtualHost` 指令中的地址都可以附带一个可选的端口。如果没有进行特别的指定，这个端口默认为主服务器中最近的一个 `Listen` 指令指定的值。特殊的端口" `*` "表示匹配所有端口。所有这一系列地址(包括由DNS查询出的所有 `A` 记录)统称虚拟主机的地址集*(address set)*。

如果没有对一个特定的IP地址使用 `NameVirtualHost` 指令，那么第一个使用这个地址的虚拟主机将被视为基于IP的虚拟主机。IP地址也可以用通配符" `*` "表示。

如果使用了基于域名的虚拟主机，那么必须用 `NameVirtualHost` 指令为这个基于域名的虚拟主机指定IP地址集。换句话说，您必须在配置文件中通过 `NameVirtualHost` 指令指定包括主机名映射(CNAME)的IP地址。

可以使用很多 `NameVirtualHost` 指令来分别对应一套 `NameVirtualHost` 指令，但对于每个特定的"IP:port"对来说，只能使用一次 `NameVirtualHost` 指令。

`NameVirtualHost` 和 `VirtualHost` 指令出现的顺序并不重要。只有对应_同一个_IP地址的 `VirtualHost` 指令的次序才是重要的。所以下面两例所起的作用是完全相同的：

```
NameVirtualHost 111.22.33.44

&lt;VirtualHost 111.22.33.44&gt;

# server A

...

&lt;/VirtualHost&gt;

&lt;VirtualHost 111.22.33.44&gt;

# server B

...

&lt;/VirtualHost&gt;

NameVirtualHost 111.22.33.55

&lt;VirtualHost 111.22.33.55&gt;

# server C

...

&lt;/VirtualHost&gt;

&lt;VirtualHost 111.22.33.55&gt;

# server D

...

&lt;/VirtualHost&gt;
```

```
&lt;VirtualHost 111.22.33.44&gt;

# server A

&lt;/VirtualHost&gt;

&lt;VirtualHost 111.22.33.55&gt;

# server C

...

&lt;/VirtualHost&gt;

&lt;VirtualHost 111.22.33.44&gt;

# server B

...

&lt;/VirtualHost&gt;

&lt;VirtualHost 111.22.33.55&gt;

# server D

...

&lt;/VirtualHost&gt;

NameVirtualHost 111.22.33.44

NameVirtualHost 111.22.33.55
```

(为了使您的配置文件更具可读性，我们推荐您使用左边的格式)

在解析完 `VirtualHost` 指令后，虚拟主机服务器将被赋予在它的 `VirtualHost` 指令中第一个名字对应的端口作为默认的 `Listen` 端口。

如果所有域名都指向同一个地址集的话， `VirtualHost` 指令中的所有域名列表都将会得到和 `ServerAlias` 指令一样的处理(但不会被其他 `ServerAlias` 语句覆盖)。请注意，这个虚拟主机自带的 `Listen` 指令将不能影响到那个地址集的端口号。

在初始化的过程中，将会为每一个IP地址产生一个列表，并插入到一个散列表中。如果这个IP地址是用在一个 `NameVirtualHost` 指令中的，这个列表将会包含所有指定为这个IP地址的基于域名的虚拟主机。如果没有虚拟主机针对这个IP地址，那么 `NameVirtualHost` 指令将被忽略，并会在日志中记录一个错误信息。对于基于IP的虚拟主机而言，这个散列表中的列表为空。

因为使用了高效的散列算法，使得在请求到达的时候在其中查找IP地址的开销变得很小，或者根本不需考虑。而且这个表格还为只有最后一个八进制位不同的IP地址做了优化。

虚拟主机的每个变量都有初始值。特别是以下这些：

1. 如果虚拟主机没有 `ServerAdmin` ， `ResourceConfig` ， `AccessConfig` ， `Timeout` ， `KeepAliveTimeout` ， `KeepAlive` ， `MaxKeepAliveRequests` ， `ReceiveBufferSize` ， `SendBufferSize` 指令，那么将从主服务器继承它们的值。(也就是说，使用在主服务器中最后出现的设定值)。

2. 虚拟主机的默认目录权限将继承主服务器的设置(包括所有模块针对每个目录的配置信息)。
3. 虚拟主机将继承主服务器中每个模块针对主服务器的设置。

本质上，主服务器在建立每个虚拟主机的时候，充当了一个默认值或根基的角色。但这些存在于主服务器中的定义的位置是无关紧要的——主服务器的配置在与虚拟主机整合之前就已经解析过了。所以即使一个主服务器的配置出现在虚拟主机定义的后面，它也同样会影响到虚拟主机的配置。

如果没有定义主服务器中的 `ServerName` ，那么将由运行这个 `httpd` 服务的机器的主机名来代替。我们将由DNS查找此 `ServerName` 返回的IP地址称为主服务器地址集*(main_server address set)*。

在没有定义 `ServerName` 的情况下，一个基于域名的虚拟主机默认采用定义虚拟主机时在 `VirtualHost` 指令中最先出现的地址。

所有使用了" `_default_` "通配符的虚拟主机将被赋予和主服务器相同的 `ServerName` 。

虚拟主机匹配

服务器用下述方法来确定对一个特定的请求使用哪个虚拟主机：

散列表查找

当客户端第一次连接的时候，会从内部的IP散列表中查找客户端想要连接的IP地址。

如果查找失败(没有找到相应的IP地址)，而所请求的端口又存在一个" `_default_` "虚拟主机，那么这个请求将会由这个虚拟主机来伺服。如果没有找到这样的" `_default_` "虚拟主机，那么这个请求将会由主服务器来伺服。

如果在散列表中没有找到IP地址，但存在一个" `NameVirtualHost *` "指令与所请求的端口号相匹配，那么将用这个虚拟主机来处理这个请求。

如果查找成功(找到了对应于这个IP地址的列表)，下一步就是看我们要处理的是一个基于IP的虚拟主机还是一个基于域名的虚拟主机。

基于IP的虚拟主机

如果返回的列表中域名列表为空，那么我们处理的就是一个基于IP的虚拟主机，这个虚拟主机将会直接进行处理而不会有其他步骤。

基于域名的虚拟主机

如果返回的域名列表包含一个或多个虚拟主机的结构，那么我们处理的就是一个基于域名的虚拟主机。这个列表包含的虚拟主机的顺序与配置文件中相应 `VirtualHost` 指令出现的顺序是相同的。

这个列表中第一个虚拟主机(也就是在配置文件中第一个指定了这个IP地址的虚拟主机)对处理请求有着最高的优先级。所有对未知服务器名或没有" `Host:` "头的请求都将由它进行处理。

如果客户端在请求中提供了一个" `Host:` "头，那么将在列表中查找第一个 `ServerName` 或 `ServerAlias` 与其符合的虚拟主机，并将其用于伺服这个请求。尽管" `Host:` "头中可以包含端口号，但Apache还是会用收到请求的那个真实端口来进行匹配。

如果客户端提交了一个不包含" `Host:` "头的HTTP/1.0的请求，我们将无法确认客户端想要连接那个服务器。而如果存在一个 `ServerPath` 与客户端提交的请求中的URI相对应，那么列表中第一个符合条件的虚拟主机将用于伺服这个请求。

如果还是找不到对应的虚拟主机，那么这个请求将会由客户端连接的IP对应的列表中的第一个与请求的端口相同的虚拟主机来伺服(如前所述)。

持久连接

上述IP查找对一个特定的TCP/IP进程只执行一次。但在持久连接(KeepAlive)中，每个请求都会进行一次这样的查找过程。换句话说，一个客户端在一个持久连接中可以向位于不同的基于域名的虚拟主机的页面提出请求。

绝对URI

如果请求提交的URI是一个绝对URI，而其中的主机名和端口号又和主服务器或某个虚拟主机相符合，并且也与作为此请求提交对象的地址和端口相符，那么这个请求的类型/主机名/端口前缀将被抹除，仅留下相对URI为对应的主服务器或虚拟主机所伺服。如果不满足上述符合条件，这个URI将保留原样，而此请求将被作为一个代理请求处理。

备忘录

- 基于域名的虚拟主机和基于IP的虚拟主机之间互相不干扰。基于IP的虚拟主机只接受发送到它自身地址集的请求，而不接受其他IP地址。基于域名的虚拟主机也是一样，它们只接受 `NameVirtualHost` 指令定义的地址集的访问。
- 永远不会对一个基于IP的虚拟主机执行 `ServerAlias` 和 `ServerPath` 检查。
- 在配置文件中，基于域名的虚拟主机、基于IP的虚拟主机、" `_default_` "虚拟主机和 `NameVirtualHost` 指令出现的顺序并不重要。而对于某个指定的地址集来说，基于域名的虚拟主机的顺序是不能混淆的：在配置文件中较先出现的虚拟主机在相应的地址集中有较高的优先权。
- 出于安全性的考虑，在" `Host:` "头中出现的端口号将不用于匹配。Apache会一直使用客

户端所连接的真实端口作为匹配。

- 如果一个 `ServerPath` 指令凑巧是后面出现的另外一个 `ServerPath` 指令的前缀，前者将用于匹配，而后者将被忽略。(这里讨论的是没有" `Host:` "头来将这两个情况分开的情况下)
- 如果有两个基于IP的虚拟主机使用了同一个地址，则在配置文件中首先出现的那个用于匹配。这种事情可能发生在你疏忽的时候。当服务器遇到这种情况的时候，会在日志文件中写入一个错误信息。
- 仅当没有其他虚拟主机符合客户端请求的IP地址和端口号时，" `_default_` "虚拟主机才会捕获这个请求。并且仅当" `_default_` "虚拟主机的端口号(默认值由您的 `Listen` 指定)与客户端发送请求的目的端口号相符时，这个请求才会被捕获。也可以使用通配符(例如：" `_default_:*` ")来捕获任何端口号的请求。这也同样适用于" `NameVirtualHost *` "的虚拟主机。
- 仅当客户端连接的目的IP地址和端口号没有指定而且不与任何一个虚拟主机(包括" `_default_` "虚拟主机)匹配的时候，才会用主服务器来伺服请求。换句话说，主服务器仅捕获没有指定IP地址和端口的请求(除非存在一个匹配端口的" `_default_` "虚拟主机)。
- 如果客户端连接到一个用于基于域名的虚拟主机使用的地址(和端口)，比如说使用了 `NameVirtualHost` 指令，那么一个未知的或没有" `Host:` "头的请求就不会与" `_default_` "虚拟主机或是主服务器相匹配。
- 绝对不能在 `VirtualHost` 指令中使用DNS名称，否则您的服务器就会依赖DNS来进行启动。而且，如果您无法控制列表中所有的域，您将会面临安全威胁。您可以在这里获得关于这个问题和以下两个问题的更多详情。
- 应当为每个虚拟主机设定 `ServerName` 。否则就会需要为每个虚拟主机进行DNS查询。

小技巧

作为DNS问题页面小技巧的附加，这里有些额外的技巧：

- 将所有主服务器的定义放在所有 `VirtualHost` 定义之前(为了增加可读性)，否则会使得类似在虚拟主机旁边的定义影响到所有的虚拟主机这样的问题不容易发现。
- 将您配置中相应的 `NameVirtualHost` 和 `VirtualHost` 定义放到一起，以获得更好的可读性。
- 避免前一个 `ServerPaths` 是后一个 `ServerPaths` 的前缀。如果您无法避免这样的情况，您最好确保在您的配置文件中"长在前，短在后"(也就是说："ServerPath/abc/def"应当出现在"ServerPath/abc"之前)。

文件描述符限制

当使用了大量虚拟主机，而且每个主机又使用了不同的日志文件时，Apache可能会遭遇文件描述符(有时也称为<cite class="calibre27">文件句柄</cite>)耗尽的困境。Apache使用的文件描述符总数如下：每个不同的错误日志文件一个、每个其他日志文件指令一个、再加10－20个作为内部使用。Unix操作系统限制了每个进程可以使用的文件描述符数量。典型上限是64个，但可以进行扩充，直至到达一个很大的硬件限制为止(hard-limit)。

尽管Apache会试着增大限制，但如果发生以下情况，则这个机制无法起作用：

1. 您的操作系统没有提供 `setrlimit()` 系统调用。
2. `setrlimit(RLIMIT_NOFILE)` 调用无法在您的系统上正常工作(比如 Solaris 2.3)
3. 文件描述符的需求量已经超出了硬件的限制。
4. 您的操作系统对文件描述符作出了其他限制。比如说限制了stdio流只能使用256以下的文件描述符。(Solaris 2)

如果遇到了这样的问题，您可以这样解决：

- 减少日志文件的数量。不在 `<VirtualHost>` 配置段中指定日志文件，而是只在主日志文件中进行记录。(参见下述分解日志文件获得详情)
- 如果您的系统因上述第1条或第2条原因不能正常工作，可以在启动Apache之前，用类似下述的脚本增大文件描述符的限制：

```
#!/bin/sh

          ulimit -S -n 100

          exec httpd
```

分解日志文件

如果您想把多个虚拟主机的日志记录到同一个日志文件中，你可能会想事后把它们分开，以对不同的虚拟主机数据进行统计分析。您可用下述方法达到这个目的。

首先，您需要将虚拟主机的信息放入日志中。您可以用 `LogFormat` 指令和" `%v` "变量达到这个目的。在您的日志格式串的开头加入它们：

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b" vhost

CustomLog logs/multiple_vhost_log vhost
```

这将用日志的普通格式来创建一个日志文件。但会在每条记录前加上正式的虚拟主机名(就是在 `ServerName` 指令中定义的那个)。(参见自定义日志格式以获取更多内容)

当您想将日志文件分开(每个虚拟主机一个日志文件)的时候，您可以使用 `split-logfile` 程序来完成这个工作。您将在Apache发行版的 `support` 目录中找到这个程序。

用如下命令来运行这个程序：

```
split-logfile < /logs/multiple_vhost_log
```

当这个程序在给予一个虚拟主机日志文件作为参数的情况下，会为日志文件中的每个虚拟主机建立一个文件。每个文件都以" `主机名.log` "这样的形式命名。

关于DNS和Apache

本文档的涵义用一句话总结就是：不要让Apache在分析配置文件的时候使用到DNS解析。如果Apache在分析配置文件时用到了DNS解析，您的服务器就会发生可靠性的问题(也可能根本无法启动)，或者遭致拒绝(偷窃)服务攻击(包括用户可以从其他用户那里偷窃点击)。

一个简单示例

```
<VirtualHost www.abc.dom>
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```

为了让Apache功能正常，一个虚拟主机绝对需要以下两部分的信息：`ServerName` 和与该服务器绑定的至少一个IP地址。上述示例没有包括IP地址，于是Apache必须要使用DNS解析来查询 `www.abc.dom` 的地址。如果在某些不可预料的情况下，当您的服务器解析配置文件时没有得到DNS的支持，那么这个虚拟主机将不会被配置。它将不会对任何请求作出反应。(在Apache1.2之前的版本，服务器甚至无法启动)。

假设 `www.abc.dom` 的IP地址是10.0.0.1，那么考虑以下这个配置片断：

```
<VirtualHost 10.0.0.1>
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```

现在Apache需要DNS对这个虚拟主机进行反向域名解析来确定其 `ServerName` 。如果反向解析失败，那么将导致这个虚拟主机功能丧失(在Apache的1.2版本之前，服务器将不能启动)。如果虚拟主机是基于域名的，它将完全不能使用，但如果它是基于IP的，那么它将很有可能工作。然而，如果Apache不得不为一个已经包含了服务器域名的服务器产生一个完整的URL，那么它将可能产生一个无效的URL。

以下是一个可以避免上述两个问题的配置片断：

```
<VirtualHost 10.0.0.1>
ServerName www.abc.dom
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```

拒绝服务

拒绝服务主要由(至少)两种形式导致。如果您在运行Apache1.2以前的版本，在上述两种情况下，如果您的任何一个虚拟主机的DNS解析失败，您都会无法启动服务。在一些情况下，DNS解析甚至不在您的控制范围之内。比如说，如果 `abc.dom` 是您的一个客户，而且他们自己控制着DNS。那么仅仅是因为他们删除了 `www.abc.dom` 这个记录，都会导致您1.2版本以前的Apache无法启动。

另外一种形式就更隐蔽了。比如说下面这个配置片断：

```
<VirtualHost www.abc.dom>
  ServerAdmin webgirl@abc.dom
  DocumentRoot /www/abc
</VirtualHost>
<VirtualHost www.def.dom>
  ServerAdmin webguy@def.dom
  DocumentRoot /www/def
</VirtualHost>
```

假设您已经为 `www.abc.dom` 设定了10.0.0.1、为 `www.def.dom` 设定了10.0.0.2。更进一步，假设 `def.dom` 自己控制DNS。在这种配置下， `def.dom` 可以将所有指向 `abc.dom` 的流量据为己。为了达到这个目的，他们只需把 `www.def.dom` 的地址解析设置成10.0.0.1就可以了。因为他们控制着自己的DNS服务，所以您无法阻止他们把 `www.def.dom` 这个记录指向任何一个IP地址。

然后，所有向10.0.0.1发出的请求(包括用户所有类似 `http://www.abc.dom/whatever` 的URL)都将会被 `def.dom` 这个虚拟主机所接收。为了更好的理解这一切是怎样发生的，您需要一个关于Apache是怎样将进入的请求分配给它的虚拟主机的深入说明。您可以在这里找到一个完整的文档。

"主服务器"地址

在Apache1.1中，基于域名的虚拟主机支持需要Apache知道运行 `httpd` 的主机的IP地址。可以用全局变量 `ServerName` (如果存在)或者调用C函数 `gethostname` (与在命令行模式下键入"hostname"得到的返回值一样)。接着它就会利用DNS来查找这个地址。目前还没有办法避免这样的查找。

如果您担心这样的查找会因为您的DNS服务器没有启动而遭到失败的结果，您就可以在 `/etc/hosts` 中插入一条记录来确定主机名(此文件中应该已经存在这条记录了，否则您的机器可能无法正常启动)。然后，要确认您的机器已经配置为当DNS解析失败的情况下会使用 `/etc/hosts` ，根据所使用的操作系统不同，您可能需要在 `/etc/resolv.conf` 或 `/etc/nsswitch.conf` 两个文件中选择一个进行编辑。

如果您的服务器不必因为其他理由而使用DNS，您也许不必在把 `HOSTRESORDER` 环境变量设为"local"的情况下运行Apache。这取决于您所使用的操作系统和解析库。如果您没有使用 `mod_env` 来控制环境变量，它还将影响到CGI。强烈建议您查看操作系统附带的man帮助或FAQ。

避免这些问题的小技巧

- 在 `VirtualHost` 中使用IP地址
- 在 `Listen` 中使用IP地址
- 确保所有的虚拟主机拥有显式的 `ServerName` 定义
- 创建一个不包含任何页面的 `<VirtualHost _default_:*>` 服务器

附录：进一步的提示

涉及到DNS的情况都很让人很不舒服。在Apache1.2中，我们努力想让服务器在DNS解析失败的情况下至少保持能够启动，但我们没能做到。在当今重编号成了必须的Internet上面，在配置文件中显式的写明IP地址已经成为不合时宜的行为了。

上述盗窃攻击的解决办法是在一个正向DNS查询后再进行一个逆向DNS解析并将两个结果进行比较。如果不同，就禁用相应的虚拟主机。这个方法需要一个正确配置了的逆向域名解析服务器(因为FTP服务器和TCP封装进行的"双重逆向"DNS处理的普遍应用，这已为大部分管理员所熟知了)。

在某些情况下，如果没有使用IP地址而DNS解析又失败了，那么正常启动一个基于域名的虚拟主机看来是不可能的。一些诸如禁用部分配置文件这样的权宜之计会带来比根本不能启动更遭的不可预测的结果。

随着HTTP/1.1的部署以及浏览器和代理服务器开始支持 `Host` 头，我们完全避免使用基于IP的虚拟主机也逐渐成为可能。这种状况下，web服务器也不必在配置时进行DNS的查询。但在1997年3月，这些特性的采用还没有广泛到可以在重要的web服务器应用的地步。

经常问到的问题

这个FAQ的最新版本总是可以从Apache主站点得到，位于 <http://httpd.apache.org/docs/2.2/faq/>

如果你的问题在这里没有找到答案，你也可以看看Apache 1.3 FAQ，看你的问题是否在那里有了答案。

主题

背景

什么是Apache ？

Apache软件基金会(ASF)是一个非营利性组织，它为Apache社区的开源软件项目提供支持。欲知详情，请查看Apache Software Foundation FAQ页面。

Apache HTTP Server(也被称为Apache httpd)是Apache软件基金会的一个创建健壮的、工业级的、功能强大的、开放源代码的HTTP(Web)服务器的项目。欲知详情，请查看About Apache页面。

什么是 Apache HTTP Server ？

- 一个强大的、灵活的、兼容HTTP/1.1规范的web服务器

- 实现了最新的协议，包括HTTP/1.1(RFC2616)
- 具有高度的可配置性和使用第三方模块的可扩展性
- 可以通过使用Apache模块API编写自己的模块进行定制
- 在非限制性许可证下提供所有的源代码
- 可以运行在 Windows 2003/XP/2000/NT/9x 、 Netware 5.x 及以上版本、 OS/2 、 大多数Unix版本以及其它操作系统上
- 被非常活跃的进行开发
- 鼓励用户反馈新想法、bug报告、补丁程序

Apache是如果进行充分测试的？

Apache正在数以百万的网络服务器上运行。它同时经过开发者和用户的充分测试。Apache HTTP Server 项目按照非常严格的标准发布服务器的新版本，并且有70%的WWW服务器在24小时不间断地运行着我们的服务器。一旦有bug被发现，我们将以最快的速度发布补丁程序和新版本。

我可以在我的产品或网站中使用Apache的logo吗？

不可以使用、复制、修改任何来自Apache软件基金会的原始图形。除非满足以下条件：

- 你可以在一个使用Apache作为web服务器的网站上使用'Powered by Apache'图标。
- 当且仅当这种使用可以促进Apache的推广时，你才可以在产品描述中使用上述'Powered by Apache'图标或Apache软件基金会logo 。严格禁止将Apache的名称或图形用于产品的签名或者服务。

支持

- "我为什么不能...？为什么...不工作？"在有问题的情况下该怎么办？
- 我要找谁寻求帮助？

"我为什么不能...？为什么...不工作？"在有问题的情况下该怎么办？

如果你使用Apache服务器软件遇到了问题，采取以下几步：

检查错误日志！

Apache服务器在遇到问题时会尽力做到对你有所帮助。在许多情况下，它会通过在错误日志中写入一条或多条消息来提供一些细节。有时这已经足够让你自己诊断和解决问题了(比如文件权限或类似的问题)。错误日志的默认位置在 `/usr/local/apache2/logs/error_log` ，但是最后还是看看配置文件中的 `ErrorLog` 指令以确认错误日志在你服务器上的确切位置。

再一次检查错误日志！

几乎所有问题都可以通过阅读错误日志来解决。

察看FAQ!

最新版本的Apache常见问题列表总是可以从Apache主站点得到。

察看Apache bug数据库

大多数报告给Apache项目组的问题都记录在bug数据库中。在你添加一个新bug之前，请务必检查已有的报告(打开的和关闭的)。如果你发现你的问题已经被报告了，请不要添加一个"我也是"那样的报告。如果原始报告还没有关闭，我们建议你经常周期性地来看看它。你也可以考虑与最初的提交者接触，因为有可能会在邮件交流中发现没有记录在数据库中的问题。

在某个用户论坛中提问

Apache拥有一个活跃的、愿意共享知识的用户社区。参与这个社区通常是获得解答的最快最好的办法。

用户邮件列表

Freenode IRC上的#apache频道也是关于用户支持的。

提交问题报告到bug数据库

如果做了以上几个合适的步骤而没有得到解答，那么请务必让httpd的开发者了解这个问题，到这里提交bug报告。

如果你的问题涉及到服务器崩溃并产生了内核dump，请在报告中包含一个backtrace(如果可能)。

我要找谁寻求帮助？

因为有数百万用户和区区不到60名志愿开发者，我们无法为Apache提供个体支持。对于免费的支持，我们建议用户参与一个用户论坛。

Apache的专业商业支持可以从许多公司得到。

错误信息

- Invalid argument: core_output_filter: writing data to the network
- AcceptEx failed
- Premature end of script headers
- Permission denied

Invalid argument: core_output_filter: writing data to the network

Apache在可能的平台上使用系统调用 `sendfile` 来加速响应的发送。不幸的是，在某些系统上，Apache会在编译时检测 `sendfile` 的存在，即使它不能正常工作。这经常发生在使用网络或其他非标准文件系统时。

这个问题的表现症状包括上述信息出现在错误日志里及对于非零长度文件请求发送零长度的响应。一般这个问题只发生在静态文件上，因为动态文件通常用不到 `sendfile` 。

要修正这个问题，可用 `EnableSendfile` 指令关闭服务器所有部分对 `sendfile` 的使用即可。同时参看 `EnableMMAP` 指令，对相似的问题有帮助。

AcceptEx Failed

如果你在win32系统上得到一个与 `AcceptEx` 系统调用相关的错误信息，参见 `Win32DisableAcceptEx` 指令。

Premature end of script headers

大多数导致这个错误的CGI脚本问题将会向浏览器发送一个" `Internal Server Error` "错误信息。要解决这种问题参见：CGI指南。

Permission denied

`error_log` 中的" `Permission denied` "错误伴随一个发送到客户端的" `Forbidden` "信息通常表明违反了文件系统的权限，而不是Apache HTTP的配置文件出了错误。检查并确认用于运行子进程的 `User` 和 `Group` 有访问导致问题的文件的足够权限。同时检查一下导致问题的文件所在的目录及其所有父目录是否具有执行(搜索)权限(也就是 `chmod +x`)。

最近发行的 Fedora Core 和其它Linux发行版使用了SELinux进行额外的访问控制，违反这些限制也会导致" `Permission denied` "消息。参见Fedora SELinux FAQ和Apache SELinux Policy Document以获得更多信息。

Apache的SSL/TLS加密

Apache HTTP服务器的 `mod_ssl` 模块提供了对使用安全套接层(Secure Sockets Layer)和传输层安全(Transport Layer Security)协议的OpenSSL库的接口。此模块和本文都是基于 Ralf S. Engelschall 的 `mod_ssl` 项目。

文档

- 简介
- 兼容性
- 如何...
- 常见问题解答
- 词汇表

mod_ssl

此模块提供的有关指令和环境变量的文档，参见mod_ssl文档。

SSL/TLS高强度加密：绪论

> 标准的好处就是你有充足的选择。如果确实不喜欢现存的标准，你只需等待来年发布一个你喜欢的新标准。
>
> -- <cite class="calibre63">A. Tanenbaum</cite>, "Introduction to Computer Networks"

作为绪论，本文针对的是熟悉Web、HTTP、Apache的读者而不是安全方面的专家，它不是SSL协议的权威性指南，不讨论在一个组织中管理证书的特殊技术，也没有重要的法定专利声明及摘录和引用限制。但是，本文会通过综合讲述各种概念、定义和例子，给 `mod_ssl` 的使用者提供背景资料，作为更深入探索的起点。

这里的内容主要是来源于Introducing SSL and Certificates using SSLeay并经过作者Frederick J. Hirsch许可。此文由 Open Group Research Institute于1997年夏，发表在Web Security: A Matter of Trust, World Wide Web Journal, Volume 2, Issue 3, Summer 1997，肯定意见请反馈给Frederick Hirsch(原作者)，反对意见请反馈给Ralf S. Engelschall(`mod_ssl` 的作者)。

密码技术

要理解SSL就必须理解密码系统、消息摘要函数(单向或散列函数)和数字签名，这些技术是许多文献所讨论的主题(比如[AC96)，提供了保密性、完整性和认证的基础。

密码系统

假设Alice想给她的银行发一个消息以划转资金，并希望这个消息是保密的，因为其中含有她的帐号和划转金额等信息。一种方案是使用密码系统，将要传输的信息转变为加密形式，从而只能为希望他读懂的人读懂。一旦加密为这种形式，这条消息也许只能用一个密钥来破译，如果没有，那么这条信息毫无用处，因为好的密码系统可以使破译难度高到入侵者认为原文不值得他们花费那么大的努力。

密码系统有两大类：常规的和公共密钥。

常规密码

又称为对称密码，需要发送者和接收者共同持有一个密钥：一小段用来加密和解密的秘密信息。如果这个密钥是保密的，那么这条消息除了发送者和接受者以外可能没有人可以阅读。如果Alice和银行共同持有一个密钥，则可以互相发送保密信息。但是，私有通讯密钥的选择行为本身，却可能不是无懈可击的。

公共密钥密码

又称为不对称密码，定义了一种使用两个密钥的算法以解决密钥交换问题，一个密钥用于加密，另一个用于解密，从而使简单公布一个密钥(公共的密钥，简称：公钥)而保留其他的(私有的密钥，简称：私钥)以接收保密消息成为可能。

任何人都可以用公钥加密一条消息，而仅允许私钥的持有者阅读。如此，Alice就可能使用公钥加密其保密消息，发送给私钥的持有者(银行)，只有银行能够对它解密。

消息摘要

虽然Alice可能加密其消息使它称为私有的，但仍应注意到某些人可能会篡改或替换其原始消息，以划转资金到他们自己的帐户。一种保证Alice消息完整性的方法是同时发一个其消息的简单摘要给银行，供银行与消息本身比对，如果相符则消息正确。

这样的方法被称为<dfn class="calibre27">消息摘要</dfn>、单向函数或散列函数。消息摘要用于对较大而且变长的消息建立较短而且等长的一种表述，其设计使将摘要还原成消息极其困难，而且对两个不同的消息几乎不可能生成相同的摘要，从而排除了替换一个消息为另一个而维持相同摘要的可能性。

Alice面临的另一个挑战是要保证摘要发送到银行的安全，如此，才能确保消息的完整性。

一种解决方法是在摘要中包含数字签名。

数字签名

当Alice发送消息到银行，银行需要确认此消息的确是她发送的，而不是入侵者盗用其帐号。为此，可以在消息中包含一个由Alice建立的数字签名。

数字签名是以加密的消息摘要和其他信息(比如一个流水号)以及发送者的私有密钥建立的。虽然任何人都可能用公共密钥解密签名，但是只有签发者知道其私有密钥，也就是，只有密钥的持有者才能签发。包含在签名中的摘要只对该消息有效，以确保没有人可以改变摘要而保持签名不变。

为了避免签名日后被入侵者破译和再利用，签名包含有一个流水号。如此，万一(只是假设)Alice并没有发送此消息，虽然她可能真的签发过，银行可以免遭其欺诈性指控。

证书

虽然Alice可能已经发送了一个保密的消息给银行，签了名，并可以保证消息的完整性，但是她仍然需要确认她的确是在和那个银行通讯，也就是说，她需要确认她用的公共密钥的确对应于银行用的私有密钥。同样，银行也需要验证此消息的签名的确对应于Alice的签名。

如果各部分有验证其余部分一致性的证书，以确认公共密钥，并是由一个可以信任的代理所签发的，那么他们双方都可以肯定其通讯对象的身份。这个可以信任的代理称为证书机构 *(Certificate Authority)*，其证书用于认证。

证书的内容

与一个公共密钥关联的证书有一个主题，即一个个体或者服务器或者其他实体的真实身份，如表1所示。主题中的信息包含身份信息(识别名[Distinguished Name])和公共密钥，还包括发布此证书的证书机构的名称和签名以及证书的有效期限，还可能会有证书机构监管者信息和流水号等附加信息。

表 1: Certificate Information

Subject	Distinguished Name, Public Key
Issuer	Distinguished Name, Signature
Period of Validity	Not Before Date, Not After Date
Administrative Information	Version, Serial Number
Extended Information	Basic Constraints, Netscape Flags, etc.

识别名用于在一个特定的上下文中指明身份，比如，一个个体可能有一个个人证书，同时还有一个其雇佣者的证书。X.509标准[X509]中定义了识别名的各个项及其名称和缩写(见表2)。

表 2: Distinguished Name Information

DN Field	Abbrev.	Description	Example
Common Name	CN	Name being certified	CN=Joe Average
Organization or Company	O	Name is associated with this organization	O=Snake Oil, Ltd.
Organizational Unit	OU	Name is associated with this organization unit, such as a department	OU=Research Institute
City/Locality	L	Name is located in this City	L=Snake City
State/Province	ST	Name is located in this State/Province	ST=Desert
Country	C	Name is located in this Country (ISO code)	C=XZ

证书机构可能会定义规定哪些识别名是可选的，而哪些是必须的一个规范，还可能对项的内容和证书使用人数有所要求。比如，一个Netscape浏览器要求证书中的Common Name项必须是服务器名称，此名称可以是服务器域名的通配模式，形如： `*.snakeoil.com` 。

证书的二进制形式用ASN.1记号法[X208] [PKCS]表示，记号法定义了如何表示内容，编码规则定义了如何将信息转变成二进制形式。证书的二进制编码使用了基于更通用的基本编码规则(Basic Encoding Rules[BER])的识别名编码规则(Distinguished Encoding Rules[DER])。为了在不能处理二进制的情况下进行传输，二进制形式用Base64编码方式[MIME]转换成ASCII形式，其编码结果是以开始和结束符号分隔的若干的行，称为PEM形式(其名称来源于"Privacy Enhanced Mail")，如下所示：

Example of a PEM-encoded certificate (snakeoil.crt)

```
-----BEGIN CERTIFICATE-----
MIIC7jCCAlegAwIBAgIBATANBgkqhkiG9w0BAQQFADCBqTELMAkGA1UEBhMCWFkx
FTATBgNVBAgTDFNuYWtlIERlc2VydDETMBEGA1UEBxMKU25ha2UgVG93bjEXMBUG
A1UEChMOU25ha2UgT2lsLCBMdGQxHjAcBgNVBAsTFUNlcnRpZmljYXRlIEF1dGhv
cml0eTEVMBMGA1UEAxMMU25ha2UgT2lsIENBMR4wHAYJKoZIhvcNAQkBFg9jYUBz
bmFrZW9pbC5kb20wHhcNOTgxMDIxMDg1ODM2WhcNOTkxMDIxMDg1ODM2WjCBpzEL
MAkGA1UEBhMCWFkxFTATBgNVBAgTDFNuYWtlIERlc2VydDETMBEGA1UEBxMKU25h
a2UgVG93bjEXMBUGA1UEChMOU25ha2UgT2lsLCBMdGQxFzAVBgNVBAsTDldlYnNl
cnZlciBUZWFtMRkwFwYDVQQDExB3d3cuc25ha2VvaWwuZG9tMR8wHQYJKoZIhvcN
AQkBFhB3d3dAc25ha2VvaWwuZG9tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDH9Ge/s2zcH+da+rPTx/DPRp3xGjHZ4GG6pCmvADIEtBtKBFAcZ64n+Dy7Np8b
vKR+yy5DGQiijsH1D/j8HlGE+q4TZ8OFk7BNBFazHxFbYI4OKMiCxdKzdif1yfaa
lWoANFlAzlSdbxeGVHoT0K+gT5w3UxwZKv2DLbCTzLZyPwIDAQABoyYwJDAPBgNV
HRMECDAGAQH/AgEAMBEGCWCGSAGG+EIBAQQEAwIAQDANBgkqhkiG9w0BAQQFAAOB
gQAZUIHAL4D09oE6Lv2k56Gp38OBDuILvwLg1v1KL8mQR+KFjghCrtpqaztZqcDt
2q2QoyulCgSzHbEGmi0EsdkPfg6mp0penssIFePYNI+/8u9HT4LuKMJX15hxBam7
dUHzICxBVC1lnHyYGjDuAMhe396lYAn8bCld1/L4NMGBCQ==
-----END CERTIFICATE-----
```

证书机构

证书机构在授予证书前会验证证书的申请信息，以确认密钥对中私有密钥的持有实体。比如：如果Alice申请一个个人证书，则证书机构首先会确认Alice的确是申请证书的那个人。

证书链

一个证书机构也可能给另一个证书机构授予证书。所以，Alice可能需要检查证书的授予者，及其父授予者，直到找到一个她所信任的。她可以只信任由一个有限的授予者链所授予的证书，以减小这个链中"劣质"证书带来的风险。

建立顶级CA

如前所述，每个证书要求其授予者指定证书主题中实体的有效性，直到最高一级的证书机构。这样就产生一个问题：最高一级的证书机构没有授予者，那么谁为它的证书作担保呢？仅在这种情况下，此证书是"自签名的"，即证书的授予者和主题中的一样，所以，必须对自签名的证书备加注意。顶级机构广泛发布的公共密钥可以减小信任这个密钥所带来的风险--这显然比其他某个人发布密钥并宣称他是证书机构要安全一些。浏览器被默认地配置为信任著名的证书机构。

许多公司是专业证书机构，如Thawte和VeriSign，提供如下服务：

- 验证证书的申请
- 处理证书的申请
- 授予和管理证书

自己建立一个证书机构也是可能的，虽然在Internet环境中有风险，但在验证个体或服务器较容易的Intranet环境中，会很有用。

证书的管理

建立一个证书机构需要一个坚强的监管、技术和管理体系。证书机构不仅仅是授予证书，还必须管理证书的有效期和更新，并维护一个已授予的但已经失效的证书列表(作废证书列表[Certificate Revocation Lists，或CRL])。比如，Alice作为公司雇员有资格申请证书，又如，Alice离开公司后需要作废此证书等。由于凭证书可以到处通行无阻，所以不可能从证书本身看出已经作废，因此，验证证书的有效性就必须查作废证书列表(而这通常不是自动处理的一部分)。

说明

如果使用了一个浏览器没有默认配置的证书机构，则必须加载这个证书机构的证书进入浏览器，使浏览器可以验证由这个证书机构签发的服务器证书。这样做是有风险的，因为一旦加载，浏览器会接受由这个证书机构签发的所有证书。

安全套接字层(SSL)

安全套接字层协议是位于可靠的面向连接的网络层协议(如TCP/IP)和应用程序协议层(如HTTP)之间的一种协议层。SSL通过互相认证、使用数字签名确保完整性、使用加密确保私密性，以实现客户端和服务器之间的安全通讯。

这个协议被设计为支持许多用于密码、摘要和签名的特定算法，允许因各种目的对特定的服务器选择算法，并允许采用新算法以得其利。其选择的协商操作发生在客户和服务器建立协议对话的开始阶段。

表 4: Versions of the SSL protocol

Version	Source	Description	Browser Support
SSL v2.0	Vendor Standard (from Netscape Corp.) [SSL2]	First SSL protocol for which implementations exists	- NS Navigator 1.x/2.x

- MS IE 3.x
- Lynx/2.8+OpenSSL | | SSL v3.0 | Expired Internet Draft (from Netscape Corp.) [SSL3] |

Revisions to prevent specific security attacks, add non-RSA ciphers, and support for certificate chains | - NS Navigator 2.x/3.x/4.x

- MS IE 3.x/4.x
- Lynx/2.8+OpenSSL | | TLS v1.0 | Proposed Internet Standard (from IETF) [TLS1] | Revision of SSL 3.0 to update the MAC layer to HMAC, add block padding for block ciphers, message order standardization and more alert messages. | - Lynx/2.8+OpenSSL |

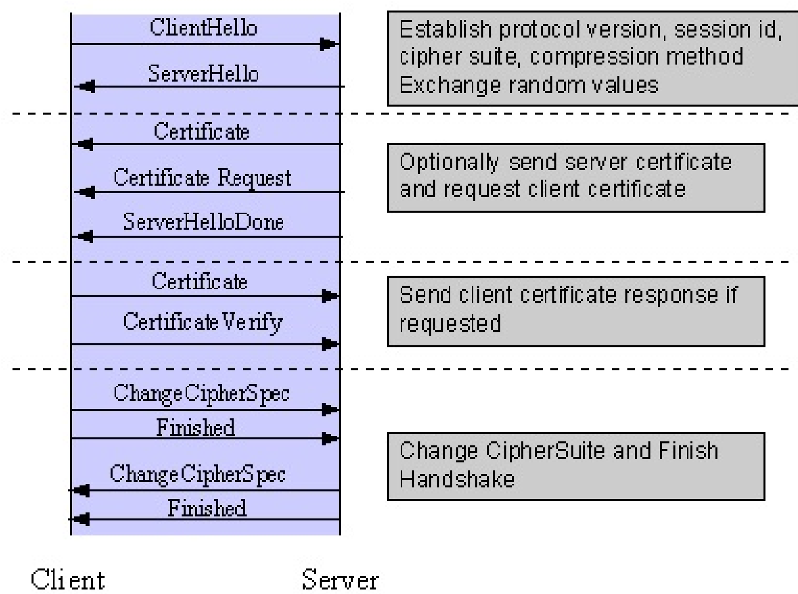
如表4所示，SSL协议有多种版本。SSL3.0的一个优点是增加了对加载证书链的支持，以允许服务器在发给浏览器的授予者证书上附加一个服务器证书。链的加载也允许浏览器验证服务器证书，即使对此授予者的证书机构证书并没有安装，因为它已经包含在这个证书链中了。SSL3.0目前正由Internet Engineering Task Force(IETF)研发，是传输层安全[TLS]协议标准的基础。

会话的建立

SSL会话在客户端和服务器的握手过程之后建立，如Figure 1所示，其过程可能因服务器是否配置为支持服务器证书和是否要求有客户证书有所不同。虽然存在密码信息管理需要额外握手操作的情况，本文只说明其中有共性的部分，参见所有可能情况下的SSL规范。

说明

SSL会话一旦建立就可能是可重用的，以避免在初始会话时的性能损失和许多步骤的重复。为此，服务器为其后的连接缓存了为每个SSL会话设定的唯一的会话标志，以减少握手操作(直到服务器缓存中的会话标志过期为止)。



<dfn class="calibre50">Figure 1</dfn>: Simplified SSL Handshake Sequence

客户端和服务器的握手过程如下所示：

1. 协商用于数据传输的密码组
2. 建立并共享客户端和服务器的会话密钥
3. 可选的客户端对服务器的认证
4. 可选的服务器对客户端的认证

第一步的密码组协商，允许客户端和服务器选择一个共同支持的密码组。SSL3.0协议规范定义了31个密码组。密码组由以下各部分组成：

- 密钥交换法
- 数据传输密码
- 建立消息认证代码(Message Authentication Code[MAC])的消息摘要

此三个组成部分说明如下。

密钥交换方法

密钥交换法指明如何在客户端和服务器的数据传输中使用共享的对称密钥。SSL2.0仅使用RSA密钥交换，而SSL3.0可以在启用证书时，选择使用包括RSA的多种密钥交换算法，以及无须证书和客户端-服务器先期通讯的Diffie-Hellman密钥交换法。

密钥交换法的一个变数是数字签名(可用可不用)，如果用，用哪一种。私有密钥配合签名可以确保在生成共享密钥[AC96, p516]的信息交换过程中抵御攻击。

数据传输密码

SSL使用在前面加密对话消息中有所讲述的常规密码算法(对称密码)，可以有包括不加密在内的九种选择：

- No encryption
- Stream Ciphers
  - RC4 with 40-bit keys
  - RC4 with 128-bit keys
- CBC Block Ciphers
  - RC2 with 40 bit key
  - DES with 40 bit key
  - DES with 56 bit key
  - Triple-DES with 168 bit key
  - Idea (128 bit key)
  - Fortezza (96 bit key)

这里的"CBC"是Cipher Block Chaining，指在加密当前块时会用到先前已经加密的部分文本；"DES"是Data Encryption Standard[AC96, ch12]，有多个变种(包括DES40和3DES_EDE)；"Idea"是现有最好的最坚强的加密算法之一；"RC2"是RSADSI[AC96, ch13]的专属的算法。

摘要函数

摘要函数指明对一个记录单元如何建立摘要。SSL有如下支持：

- No digest (Null choice)
- MD5, a 128-bit hash
- Secure Hash Algorithm (SHA-1), a 160-bit hash

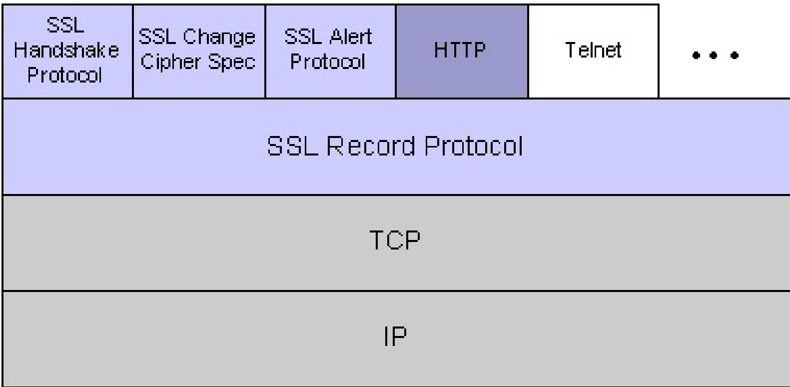
消息摘要用于建立加密的消息认证码(MAC)，与消息本身一同发送，以确保消息完整性并抵御还原攻击。

握手序列协议

握手序列使用三个协议：

- <dfn class="calibre27">SSL Handshake Protocol</dfn>，以完成客户端和服务器之间对话的建立。
- <dfn class="calibre27">SSL Change Cipher Spec Protocol</dfn>，以实际建立对话用密码组的约定。
- <dfn class="calibre27">SSL Alert Protocol</dfn>，在客户端和服务器之间传输SSL出错消息。

这些协议和应用协议的数据用 <dfn class="calibre27">SSL Record Protocol</dfn> 进行封装，如Figure 2所示，在不检查数据的较底层的协议中传输。封装协议对其底层协议来说是未知的。

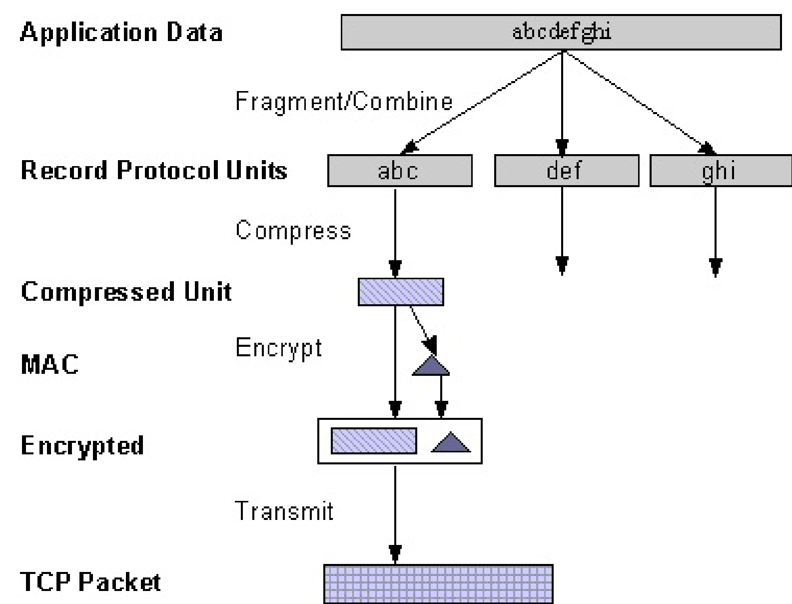


<dfn class="calibre50">Figure 2</dfn>: SSL Protocol Stack

SSL控制协议对记录协议的封装，使一个正在进行的对话在重协商其控制协议后得以安全地进行传输。如果事先没有建立对话，则会使用Null密码组，也就是说，在建立对话以前，不使用密码，且消息没有完整性摘要。

数据传输

SSL记录协议，如Figure 3所示，用于客户端和服务器之间的传输应用和SSL控制数据，可能把数据分割成较小的单元，或者组合多个较高层协议数据为一个单元。在使用底层可靠传输协议传输数据单元之前，它可能会对这些单元进行压缩、附着摘要签名和加密(注意：目前所有主要SSL的实现都缺乏对压缩的支持)。



<dfn class="calibre50">Figure 3</dfn>: SSL Record Protocol

保护HTTP通讯

SSL的一个常见的用途是保护浏览器和网络服务器之间的网络HTTP通讯，但这并排除应用于不加保护的HTTP。其方法主要是，对普通HTTP加以SSL保护(称为HTTPS)，但有一个重要的区别：它使用URL类型 `https` 而不是 `http` ，而且使用不同的服务器端口(默认的是443)。 `mod_ssl` 为Apache网络服务器提供的功能主要就是这些了...

References

[AC96]

Bruce Schneier, <q class="calibre27">Applied Cryptography</q>, 2nd Edition, Wiley, 1996. See http://www.counterpane.com/ for various other materials by Bruce Schneier.

[X208]

ITU-T Recommendation X.208, <q class="calibre27">Specification of Abstract Syntax Notation One (ASN.1)</q>, 1988. See for instance http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-X.208-198811-I.

[X509]

ITU-T Recommendation X.509, <q class="calibre27">The Directory - Authentication Framework</q>. See for instance http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.509.

[PKCS]

<q class="calibre27">Public Key Cryptography Standards (PKCS)</q>, RSA Laboratories Technical Notes, See http://www.rsasecurity.com/rsalabs/pkcs/.

[MIME]

N. Freed, N. Borenstein, <q class="calibre27">Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies</q>, RFC2045. See for instance http://ietf.org/rfc/rfc2045.txt.

[SSL2]

Kipp E.B. Hickman, <q class="calibre27">The SSL Protocol</q>, 1995. See http://www.netscape.com/eng/security/SSL_2.html.

[SSL3]

Alan O. Freier, Philip Karlton, Paul C. Kocher, <q class="calibre27">The SSL Protocol Version 3.0</q>, 1996. See http://www.netscape.com/eng/ssl3/draft302.txt.

[TLS1]

Tim Dierks, Christopher Allen, <q class="calibre27">The TLS Protocol Version 1.0</q>, 1999. See http://ietf.org/rfc/rfc2246.txt.

SSL/TLS高强度加密：兼容性

> 所有PC都是兼容的。但是其中一些比另一些更兼容。
>
> -- <cite class="calibre63">无名氏</cite>

本文讨论对其他SSL方案的向下兼容性。mod_ssl并不是Apache唯一存在的SSL方案，另外还有四种主要的产品：Ben Laurie的免费的Apache-SSL(出现在1998年，与mod_ssl同源)，RedHat商业化的Secure Web Server(基于mod_ssl)，Covalent商业化的Raven SSL Module(同样基于mod_ssl)和C2Net的商业化产品Stronghold(直到Stringhold2.x都基于一个不同的演化分支Sioux，从Stronghold3.x起基于mod_ssl)。

使用mod_ssl的原因是，mod_ssl几乎提供了在大多数情况下能够兼容其他方案的功能的超集。事实上，兼容性包括三个方面：配置指令、环境变量和自定义日志功能。

配置指令

为了兼容SSL方案的配置指令，我们做了一个简单的对应：有直接对应的指令则简单对应，没有直接对应的指令则会在日志文件中产生警告信息。表1列出已实现对应的指令。目前仅对Apache-SSL1.x和mod_ssl2.0.x有完整的向下兼容支持，而仅支持Sioux1.x和Stronghold2.x的一部分，由于其接口中的特殊功能mod_ssl目前尚不支持。

表1: 配置指令的对应

旧指令	**mod_ssl指令**	
---	---	---
Apache-SSL 1.x & mod_ssl 2.0.x 兼容性:		

`SSLEnable`	`SSLEngine on`	已
`SSLDisable`	`SSLEngine off`	已
`SSLLogFile` *file*	`SSLLog` *file*	已
`SSLRequiredCiphers` *spec*	`SSLCipherSuite` *spec*	被
`SSLRequireCipher` *c1* ...	`SSLRequire %{SSL_CIPHER} in {"` *c1* `", ...}`	无
`SSLBanCipher` *c1* ...	`SSLRequire not (%{SSL_CIPHER} in {"` *c1* `", ...})`	无
`SSLFakeBasicAuth`	`SSLOptions +FakeBasicAuth`	被
`SSLCacheServerPath` *dir*	-	已

`SSLCacheServerPort` *integer*	-	已
Apache-SSL 1.x 兼容性:		

`SSLExportClientCertificates`	`SSLOptions +ExportCertData`	被
`SSLCacheServerRunDir` *dir*	-	不
Sioux 1.x 兼容性:		

`SSL_CertFile` *file*	`SSLCertificateFile` *file*	被
`SSL_KeyFile` *file*	`SSLCertificateKeyFile` *file*	被
`SSL_CipherSuite` *arg*	`SSLCipherSuite` *arg*	被
`SSL_X509VerifyDir` *arg*	`SSLCACertificatePath` *arg*	被
`SSL_Log` *file*	`SSLLogFile` *file*	被
`SSL_Connect` *flag*	`SSLEngine` *flag*	被
`SSL_ClientAuth` *arg*	`SSLVerifyClient` *arg*	被
`SSL_X509VerifyDepth` *arg*	`SSLVerifyDepth` *arg*	被
`SSL_FetchKeyPhraseFrom` *arg*	-	没 用 SS
`SSL_SessionDir` *dir*	-	没 用 SS
`SSL_Require` *expr*	-	没 用
`SSL_CertFileType` *arg*	-	不
`SSL_KeyFileType` *arg*	-	不
`SSL_X509VerifyPolicy` *arg*	-	不
`SSL_LogX509Attributes` *arg*	-	不
Stronghold 2.x 兼容性:		

`StrongholdAccelerator` *dir*	-	不
`StrongholdKey` *dir*	-	不
`StrongholdLicenseFile` *dir*	-	不
`SSLFlag` *flag*	`SSLEngine` *flag*	被

SSLSessionLockFile *file*	SSLMutex *file*	被更
SSLCipherList *spec*	SSLCipherSuite *spec*	被更
RequireSSL	SSLRequireSSL	被更
SSLErrorFile *file*	-	不再
SSLRoot *dir*	-	不再
SSL_CertificateLogDir *dir*	-	不再
AuthCertDir *dir*	-	不再
SSL_Group *name*	-	不再
SSLProxyMachineCertPath *dir*	-	不再
SSLProxyMachineCertFile *file*	-	不再
SSLProxyCACertificatePath *dir*	-	不再
SSLProxyCACertificateFile *file*	-	不再
SSLProxyVerifyDepth *number*	-	不再
SSLProxyCipherList *spec*	-	不再

环境变量

当使用" SSLOptions +CompatEnvVars "时，会产生附加的、对应于现存官方mod_ssl变量的环境变量。表2列出了已实现的变量的演变。

表2: 环境变量的演变

旧变量	**mod_ssl 变量**	说明
---	---	---
SSL_PROTOCOL_VERSION	SSL_PROTOCOL	被更名
SSLEAY_VERSION	SSL_VERSION_LIBRARY	被更名
HTTPS_SECRETKEYSIZE	SSL_CIPHER_USEKEYSIZE	被更名
HTTPS_KEYSIZE	SSL_CIPHER_ALGKEYSIZE	被更名
HTTPS_CIPHER	SSL_CIPHER	被更名
HTTPS_EXPORT	SSL_CIPHER_EXPORT	被更名
SSL_SERVER_KEY_SIZE	SSL_CIPHER_ALGKEYSIZE	被更名
SSL_SERVER_CERTIFICATE	SSL_SERVER_CERT	被更名

SSL_SERVER_CERT_START	SSL_SERVER_V_START	被更名
SSL_SERVER_CERT_END	SSL_SERVER_V_END	被更名
SSL_SERVER_CERT_SERIAL	SSL_SERVER_M_SERIAL	被更名
SSL_SERVER_SIGNATURE_ALGORITHM	SSL_SERVER_A_SIG	被更名
SSL_SERVER_DN	SSL_SERVER_S_DN	被更名
SSL_SERVER_CN	SSL_SERVER_S_DN_CN	被更名
SSL_SERVER_EMAIL	SSL_SERVER_S_DN_Email	被更名
SSL_SERVER_O	SSL_SERVER_S_DN_O	被更名
SSL_SERVER_OU	SSL_SERVER_S_DN_OU	被更名
SSL_SERVER_C	SSL_SERVER_S_DN_C	被更名
SSL_SERVER_SP	SSL_SERVER_S_DN_SP	被更名
SSL_SERVER_L	SSL_SERVER_S_DN_L	被更名
SSL_SERVER_IDN	SSL_SERVER_I_DN	被更名
SSL_SERVER_ICN	SSL_SERVER_I_DN_CN	被更名
SSL_SERVER_IEMAIL	SSL_SERVER_I_DN_Email	被更名
SSL_SERVER_IO	SSL_SERVER_I_DN_O	被更名
SSL_SERVER_IOU	SSL_SERVER_I_DN_OU	被更名
SSL_SERVER_IC	SSL_SERVER_I_DN_C	被更名
SSL_SERVER_ISP	SSL_SERVER_I_DN_SP	被更名
SSL_SERVER_IL	SSL_SERVER_I_DN_L	被更名
SSL_CLIENT_CERTIFICATE	SSL_CLIENT_CERT	被更名
SSL_CLIENT_CERT_START	SSL_CLIENT_V_START	被更名
SSL_CLIENT_CERT_END	SSL_CLIENT_V_END	被更名
SSL_CLIENT_CERT_SERIAL	SSL_CLIENT_M_SERIAL	被更名
SSL_CLIENT_SIGNATURE_ALGORITHM	SSL_CLIENT_A_SIG	被更名
SSL_CLIENT_DN	SSL_CLIENT_S_DN	被更名
SSL_CLIENT_CN	SSL_CLIENT_S_DN_CN	被更名
SSL_CLIENT_EMAIL	SSL_CLIENT_S_DN_Email	被更名
SSL_CLIENT_O	SSL_CLIENT_S_DN_O	被更名
SSL_CLIENT_OU	SSL_CLIENT_S_DN_OU	被更名
SSL_CLIENT_C	SSL_CLIENT_S_DN_C	被更名
SSL_CLIENT_SP	SSL_CLIENT_S_DN_SP	被更名

SSL_CLIENT_L	SSL_CLIENT_S_DN_L	被更名
SSL_CLIENT_IDN	SSL_CLIENT_I_DN	被更名
SSL_CLIENT_ICN	SSL_CLIENT_I_DN_CN	被更名
SSL_CLIENT_IEMAIL	SSL_CLIENT_I_DN_Email	被更名
SSL_CLIENT_IO	SSL_CLIENT_I_DN_O	被更名
SSL_CLIENT_IOU	SSL_CLIENT_I_DN_OU	被更名
SSL_CLIENT_IC	SSL_CLIENT_I_DN_C	被更名
SSL_CLIENT_ISP	SSL_CLIENT_I_DN_SP	被更名
SSL_CLIENT_IL	SSL_CLIENT_I_DN_L	被更名
SSL_EXPORT	SSL_CIPHER_EXPORT	被更名
SSL_KEYSIZE	SSL_CIPHER_ALGKEYSIZE	被更名
SSL_SECKEYSIZE	SSL_CIPHER_USEKEYSIZE	被更名
SSL_SSLEAY_VERSION	SSL_VERSION_LIBRARY	被更名
SSL_STRONG_CRYPTO	-	mod_ssl不支持
SSL_SERVER_KEY_EXP	-	mod_ssl不支持
SSL_SERVER_KEY_ALGORITHM	-	mod_ssl不支持
SSL_SERVER_KEY_SIZE	-	mod_ssl不支持
SSL_SERVER_SESSIONDIR	-	mod_ssl不支持
SSL_SERVER_CERTIFICATELOGDIR	-	mod_ssl不支持
SSL_SERVER_CERTFILE	-	mod_ssl不支持
SSL_SERVER_KEYFILE	-	mod_ssl不支持
SSL_SERVER_KEYFILETYPE	-	mod_ssl不支持
SSL_CLIENT_KEY_EXP	-	mod_ssl不支持
SSL_CLIENT_KEY_ALGORITHM	-	mod_ssl不支持
SSL_CLIENT_KEY_SIZE	-	mod_ssl不支持

自定义日志功能

如果mod*ssl*被静态编译进*Apache*或者被动态加载*(以DSO*方式*)*，则可以使用参考文档中说明的由 `mod_log_config` 提供的自定义日志格式。但是为了向下兼容，不能使用用于扩展任何模块中任何变量的扩展格式" `%{` *_varname* `}x` "和附加的密码格式" `%{` *name* `}c` "。表3列出了已实现的格式。

表 3: 自定义日志加密格式

Function Call	格式说明
`%...{version}c`	SSL协议版本
`%...{cipher}c`	SSL密码
`%...{subjectdn}c`	客户证书的 Subject Distinguished Name
`%...{issuerdn}c`	客户证书的 Issuer Distinguished Name
`%...{errcode}c`	客户证书的出错代码(数值)
`%...{errstr}c`	客户证书的出错信息(文字)

SSL/TLS高强度加密：如何...?

> 这个问题的解决方案是简单而且直接的，只是为了给读者做做练习。
>
> -- <cite class="calibre63">标准教科书</cite>

由于SSL、HTTP、Apache三者共同对请求进行处理，这使得在支持SSL的web服务器上实现特殊的安全制约变得不那么简单。本节介绍了普通情况下的解决方案，作为找出最终方案的第一步。采用这些方案以前，先要尽量地去理解，不了解其限制和相关性就贸然使用是最糟糕的了。

加密方案和强制性高等级安全

- 仅使用SSLv2的服务器
- 仅接受高强度加密请求的服务器
- 以服务器为网关的加密
- 更强的针对目录的加密需求

如何建立一个仅使用SSLv2的服务器？

可以这样建立一个仅使用SSLv2协议及其密码算法的服务器：

httpd.conf

```
SSLProtocol -all +SSLv2

SSLCipherSuite SSLv2:+HIGH:+MEDIUM:+LOW:+EXP
```

如何建立一个仅接受高强度加密请求的SSL服务器？

如下设置为仅使用最强的七种密码算法：

httpd.conf

```
SSLProtocol all

SSLCipherSuite HIGH:MEDIUM
```

如何建立一个仅接受高强度加密请求的SSL服务器，而又允许对外浏览器使用更强的加密？

这个功能被称为以服务器为网关的加密(Server Gated Cryptography[SGC])，在随mod_ssl发布的 `README.GlobalID` 文档中有详细说明。简单地说就是：服务器拥有一个由来自Verisign的一个特殊的CA证书签发的服务器身份证，从而在对外浏览器上实现高强度加密。其过程如下：浏览器使用对外密码进行连接，服务器返回其全局ID身份证，浏览器校验后在后继HTTP通讯产生之前提升其密码组。现在的问题是：如何允许这样的提升，而又强制性地使用高强度加密。换句话说就是：浏览器必须在开始连接时就使用高强度加密，或者提升到高强度加密，但是维持对外密码是不允许的。以下巧妙地解决了这个问题：

httpd.conf

```
# 允许在初始握手阶段使用所有的密码，以允许对外服务器通过SGC功能提升密码组
SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
<Directory /usr/local/apache2/htdocs>
# 但是最终会拒绝所有没有提升密码组的浏览器
SSLRequire %{SSL_CIPHER_USEKEYSIZE} >= 128
</Directory>
```

如何建立接受所有类型密码的**SSL**服务器，但对特定的**URL**实施高强度加密？

显然，不能使用服务器全局设置 `SSLCipherSuite` ，它会限制密码为强类型。但是，mod_ssl允许重配置针对目录的密码组，并自动进行一个带有服从新配置的SSL参数的重协商。因此，其解决方案成了：

```
# 在一般情况下的处理是宽松的
SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
<Location /strong/area>
# 但对于 https://hostname/strong/area/ 及其以下的内容要求高强度密码
SSLCipherSuite HIGH:MEDIUM
</Location>
```

客户认证和访问控制

- 简单的基于证书的客户认证
- 选择性的基于证书的客户认证
- 特殊的基于证书的客户认证
- intranet和internet的认证

在知道所有客户的情况下，如何实现基于证书的客户认证？

如果你了解你的用户群体(比如：一个封闭的用户组)，正如在一个Intranet中，则可以使用一般的证书认证。所有要做的事情只是，建立由你自己的CA证书签发的客户证书 `ca.crt` ，并依此证书校验客户。

httpd.conf

```
# require a client certificate which has to be directly
# signed by our CA certificate in ca.crt
SSLVerifyClient require
SSLVerifyDepth 1
SSLCACertificateFile conf/ssl.crt/ca.crt
```

如何针对一个特定的URL对客户实施基于证书的认证，而同时又允许任何客户访问服务器其余部分？

这又要用到 `mod_ssl` 提供的针对目录的重配置功能：

httpd.conf

```
SSLVerifyClient none
SSLCACertificateFile conf/ssl.crt/ca.crt
<Location /secure/area>
SSLVerifyClient require
SSLVerifyDepth 1
</Location>
```

如何针对某些URL对特定的客户实施基于证书的认证，而同时又允许任何客户访问服务器其余部分？

其关键在于对客户证书的各个组成部分进行验证，一般就是指验证 Distinguished Name (DN)的全部或部分。有基于 `mod_auth_basic` 和基于 `SSLRequire` 类型的两种方法以验证。第一种方法适合用于客户完全属于不同类型，并为所有客户建立了密码数据库的情形；第二种方法适用于客户都属于一个被编码写入DN的公共分级的一部分的情形，因为匹配客户会更容易。

第一种方法：

httpd.conf

```
SSLVerifyClient      none
<Directory /usr/local/apache2/htdocs/secure/area>

SSLVerifyClient      require
SSLVerifyDepth       5
SSLCACertificateFile conf/ssl.crt/ca.crt
SSLCACertificatePath conf/ssl.crt
SSLOptions           +FakeBasicAuth
SSLRequireSSL
AuthName             "Snake Oil Authentication"
AuthType             Basic
AuthBasicProvider    file
AuthUserFile         /usr/local/apache2/conf/httpd.passwd
require              valid-user
</Directory>
```

httpd.passwd

```
/C=DE/L=Munich/O=Snake Oil, Ltd./OU=Staff/CN=Foo:xxj31ZMTZzkVA
/C=US/L=S.F./O=Snake Oil, Ltd./OU=CA/CN=Bar:xxj31ZMTZzkVA
/C=US/L=L.A./O=Snake Oil, Ltd./OU=Dev/CN=Quux:xxj31ZMTZzkVA
```

第二种方法：

httpd.conf

```
SSLVerifyClient      none
<Directory /usr/local/apache2/htdocs/secure/area>

  SSLVerifyClient      require
  SSLVerifyDepth       5
  SSLCACertificateFile conf/ssl.crt/ca.crt
  SSLCACertificatePath conf/ssl.crt
  SSLOptions           +FakeBasicAuth
  SSLRequireSSL
  SSLRequire       %{SSL_CLIENT_S_DN_O}  eq "Snake Oil, Ltd." \
               and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"}
</Directory>
```

如何要求来自Internet的客户使用强密码的HTTPS，并对其访问Intranet站点的子区域实施或者基本的或者客户证书的认证，而同时又允许Intranet的客户进行普通的HTTP访问？

假设Intranet客户的IP地址是192.160.1.0/24，Intranet站点子区域的URL是 `/subarea` ，则可以在HTTPS虚拟主机以外这样配置(以同时作用于HTTPS和HTTP)：

httpd.conf

```
SSLCACertificateFile conf/ssl.crt/company-ca.crt

<Directory /usr/local/apache2/htdocs>
# subarea以外的区域只允许来自Intranet的访问
Order                deny,allow
Deny                 from all
Allow                from 192.168.1.0/24
</Directory>

<Directory /usr/local/apache2/htdocs/subarea>
# 在subarea以内，允许所有来自Intranet的访问，
# 但对来自Internet的访问，仅允许HTTPS+Strong-Cipher+Password
# 或者HTTPS+Strong-Cipher+Client-Certificate

# 如果使用了HTTPS，则确保使用高强度加密
# 同时允许客户以基本认证的形式认证
SSLVerifyClient      optional
SSLVerifyDepth       1
SSLOptions           +FakeBasicAuth +StrictRequire
SSLRequire           %{SSL_CIPHER_USEKEYSIZE} >= 128

# 强制来自Internet的客户使用HTTPS
RewriteEngine        on
RewriteCond          %{REMOTE_ADDR} !^192\.168\.1\.[0-9]+$
RewriteCond          %{HTTPS} !=on
RewriteRule          .* - [F]

# 允许网络访问和基本认证
Satisfy              any

# 控制网络访问
Order                deny,allow
Deny                 from all
Allow                192.168.1.0/24

# HTTP基本认证
AuthType             basic
AuthName             "Protected Intranet Area"
AuthBasicProvider    file
AuthUserFile         conf/protected.passwd
Require              valid-user
</Directory>
```

SSL/TLS Strong Encryption: FAQ

> The wise man doesn't give the right answers, he poses the right questions.
>
> -- <cite class="calibre63">Claude Levi-Strauss</cite>

This chapter is a collection of frequently asked questions (FAQ) and corresponding answers following the popular USENET tradition. Most of these questions occurred on the Newsgroup `comp.infosystems.www.servers.unix` or the mod_ssl Support Mailing List `modssl-users@modssl.org`. They are collected at this place to avoid answering the same questions over and over.

Please read this chapter at least once when installing mod_ssl or at least search for your problem here before submitting a problem report to the author.

About The Module

- What is the history of mod_ssl?
- mod_ssl and Year 2000?
- mod_ssl and Wassenaar Arrangement?

What is the history of mod_ssl?

The mod_ssl v1 package was initially created in April 1998 by Ralf S. Engelschall via porting Ben Laurie's Apache-SSL 1.17 source patches for Apache 1.2.6 to Apache 1.3b6. Because of conflicts with Ben Laurie's development cycle it then was re-assembled from scratch for Apache 1.3.0 by merging the old mod_ssl 1.x with the newer Apache-SSL 1.18. From this point on mod_ssl lived its own life as mod_ssl v2. The first publicly released version was mod_ssl 2.0.0 from August 10th, 1998.

After US export restrictions on cryptographic software were loosened, `mod_ssl` became part of the Apache HTTP Server with the release of Apache httpd 2.

Is mod_ssl affected by the Wassenaar Arrangement?

First, let us explain what <dfn class="calibre27">Wassenaar</dfn> and its <dfn class="calibre27">Arrangement on Export Controls for Conventional Arms and Dual-Use Goods and Technologies</dfn> is: This is a international regime, established in 1995, to

control trade in conventional arms and dual-use goods and technology. It replaced the previous <dfn class="calibre27">CoCom</dfn> regime. Further details on both the Arrangement and its signatories are available at http://www.wassenaar.org/.

In short, the aim of the Wassenaar Arrangement is to prevent the build up of military capabilities that threaten regional and international security and stability. The Wassenaar Arrangement controls the export of cryptography as a dual-use good, that is, something that has both military and civilian applications. However, the Wassenaar Arrangement also provides an exemption from export controls for mass-market software and free software.

In the current Wassenaar <cite class="calibre27">List of Dual Use Goods and Technologies And Munitions</cite>, under <q class="calibre27">GENERAL SOFTWARE NOTE (GSN) </q> it says <q class="calibre27">The Lists do not control "software" which is either: 1. [...] 2. "in the public domain".</q> And under <q class="calibre27">DEFINITIONS OF TERMS USED IN THESE LISTS</q> we find <q class="calibre27">In the public domain</q> defined as <q class="calibre27">"technology" or "software" which has been made available without restrictions upon its further dissemination. Note: Copyright restrictions do not remove "technology" or "software" from being "in the public domain".</q>

So, both mod_ssl and OpenSSL are <q class="calibre27">in the public domain</q> for the purposes of the Wassenaar Arrangement and its <q class="calibre27">List of Dual Use Goods and Technologies And Munitions List</q>, and thus not affected by its provisions.

Installation

- Why do I get permission errors related to SSLMutex when I start Apache?
- Why does mod_ssl stop with the error "Failed to generate temporary 512 bit RSA private key", when I start Apache?

Why do I get permission errors related to SSLMutex when I start Apache?

Errors such as
" `mod_ssl: Child could not open SSLMutex lockfile /opt/apache/logs/ssl_mutex.18332 (System e`
" are usually caused by overly restrictive permissions on the *parent* directories. Make sure that all parent directories (here `/opt` , `/opt/apache` 和 `/opt/apache/logs`) have the x-bit set for, at minimum, the UID under which Apache's children are running (see the `User` directive).

Why does mod_ssl stop with the error "Failed to generate temporary 512 bit RSA private key", when I start Apache?

Cryptographic software needs a source of unpredictable data to work correctly. Many open source operating systems provide a "randomness device" that serves this purpose (usually named `/dev/random`). On other systems, applications have to seed the OpenSSL Pseudo Random Number Generator (PRNG) manually with appropriate data before generating keys or performing public key encryption. As of version 0.9.5, the OpenSSL functions that need randomness report an error if the PRNG has not been seeded with at least 128 bits of randomness.

To prevent this error, `mod_ssl` has to provide enough entropy to the PRNG to allow it to work correctly. This can be done via the `SSLRandomSeed` directives.

Configuration

- Is it possible to provide HTTP and HTTPS from the same server?
- Which port does HTTPS use?
- How do I speak HTTPS manually for testing purposes?
- Why does the connection hang when I connect to my SSL-aware Apache server
- Why do I get "Connection Refused" errors, when trying to access my newly installed Apache+mod_ssl server via HTTPS?
- Why are the `SSL_XXX` variables not available to my CGI & SSI scripts?
- How can I switch between HTTP and HTTPS in relative hyperlinks?

Is it possible to provide HTTP and HTTPS from the same server?

Yes. HTTP and HTTPS use different server ports (HTTP binds to port 80, HTTPS to port 443), so there is no direct conflict between them. You can either run two separate server instances bound to these ports, or use Apache's elegant virtual hosting facility to create two virtual servers over one instance of Apache - one responding to requests on port 80 and speaking HTTP and the other responding to requests on port 443 speaking HTTPS.

Which port does HTTPS use?

You can run HTTPS on any port, but the standards specify port 443, which is where any HTTPS compliant browser will look by default. You can force your browser to look on a different port by specifying it in the URL like this (for port 666):

```
https://secure.server.dom:666/
```

How do I speak HTTPS manually for testing purposes?

While you usually just use

```
$ telnet localhost 80
    GET / HTTP/1.0
```

for simple testing of Apache via HTTP, it's not so easy for HTTPS because of the SSL protocol between TCP and HTTP. With the help of OpenSSL's `s_client` command, however, you can do a similar check for HTTPS:

```
$ openssl s_client -connect localhost:443 -state -debug
    GET / HTTP/1.0
```

Before the actual HTTP response you will receive detailed information about the SSL handshake. For a more general command line client which directly understands both HTTP and HTTPS, can perform GET and POST operations, can use a proxy, supports byte ranges, etc. you should have a look at the nifty cURL tool. Using this, you can check that Apache is responding correctly on ports 80 and 443 as follows:

```
$ curl http://localhost/
    $ curl https://localhost/
```

Why does the connection hang when I connect to my SSL-aware Apache server?

Because you connected with HTTP to the HTTPS port, i.e. you used an URL of the form " `http://` " instead of " `https://` ". This also happens the other way round when you connect via HTTPS to a HTTP port, i.e. when you try to use " `https://` " on a server that doesn't support SSL (on this port). Make sure you are connecting to a virtual server that supports SSL, which is probably the IP associated with your hostname, not localhost (127.0.0.1).

Why do I get "Connection Refused" messages, when trying to access my newly installed Apache+mod_ssl server via HTTPS?

This can happen for various reasons. The most common mistakes include starting Apache with just `apachectl start` (or `httpd`) instead of `apachectl startssl` (or `httpd -DSSL`). Your configuration may also be incorrect. Please make sure that your `Listen` directives match your `<VirtualHost>` directives. If all else fails, please start afresh, using the default configuration provided by `mod_ssl` .

Why are the `SSL_XXX` variables not available to my CGI & SSI scripts?

Please make sure you have " `SSLOptions +StdEnvVars` " enabled for the context of your CGI/SSI requests.

How can I switch between HTTP and HTTPS in relative hyperlinks?

Usually, to switch between HTTP and HTTPS, you have to use fully-qualified hyperlinks (because you have to change the URL scheme). Using `mod_rewrite` however, you can manipulate relative hyperlinks, to achieve the same effect.

```
RewriteEngine on
RewriteRule   ^/(.*):SSL$   https://%{SERVER_NAME}/$1 [R,L]
RewriteRule   ^/(.*):NOSSL$ http://%{SERVER_NAME}/$1  [R,L]
```

This rewrite ruleset lets you use hyperlinks of the form `<a href="document.html:SSL">` , to switch to HTTPS in a relative link.

Certificates

certificate?

What are RSA Private Keys, CSRs and Certificates?

An RSA private key file is a digital file that you can use to decrypt messages sent to you. It has a public component which you distribute (via your Certificate file) which allows people to encrypt those messages to you.

A Certificate Signing Request (CSR) is a digital file which contains your public key and your name. You send the CSR to a Certifying Authority (CA), who will convert it into a real Certificate, by signing it.

A Certificate contains your RSA public key, your name, the name of the CA, and is digitally signed by the CA. Browsers that know the CA can verify the signature on that Certificate, thereby obtaining your RSA public key. That enables them to send messages which only you can decrypt.

See the 简介 chapter for a general description of the SSL protocol.

Is there a difference on startup between the original Apache and an SSL-aware Apache?

Yes. In general, starting Apache with `mod_ssl` built-in is just like starting Apache without it. However, if you have a passphrase on your SSL private key file, a startup dialog will pop up which asks you to enter the pass phrase.

Having to manually enter the passphrase when starting the server can be problematic - for example, when starting the server from the system boot scripts. In this case, you can follow the steps below to remove the passphrase from your private key.

How do I create a self-signed SSL Certificate for testing purposes?

1. Make sure OpenSSL is installed and in your `PATH` .
2. Run the following command, to create `server.key` 和 `server.crt` files: `**$ openssl req -new -x509 -nodes -out server.crt -keyout server.key**` These can be used as follows in your `httpd.conf` file:

   ```
               SSLCertificateFile    /path/to/this/server.crt
               SSLCertificateKeyFile /path/to/this/server.key
   ```

3. It is important that you are aware that this `server.key` does *not* have any passphrase. To add a passphrase to the key, you should run the following command, and enter & verify the passphrase as requested.

   ```
   **$ openssl rsa -des3 -in server.key -out server.key.new**
   **$ mv server.key.new server.key**
   ```

   Please backup the `server.key` file, and the passphrase you entered, in a secure location.

How do I create a real SSL Certificate?

Here is a step-by-step description:

1. Make sure OpenSSL is installed and in your `PATH` .

2. Create a RSA private key for your Apache server (will be Triple-DES encrypted and PEM formatted):

   ```
   **$ openssl genrsa -des3 -out server.key 1024**
   ```

   Please backup this `server.key` file and the pass-phrase you entered in a secure location. You can see the details of this RSA private key by using the command:

   ```
   **$ openssl rsa -noout -text -in server.key**
   ```

   If necessary, you can also create a decrypted PEM version (not recommended) of this RSA private key with:

   ```
   **$ openssl rsa -in server.key -out server.key.unsecure**
   ```

3. Create a Certificate Signing Request (CSR) with the server RSA private key (output will be PEM formatted):

   ```
   **$ openssl req -new -key server.key -out server.csr**
   ```

   Make sure you enter the FQDN ("Fully Qualified Domain Name") of the server when OpenSSL prompts you for the "CommonName", i.e. when you generate a CSR for a website which will be later accessed via `https://www.foo.dom/` , enter "www.foo.dom" here. You can see the details of this CSR by using

   ```
   **$ openssl req -noout -text -in server.csr**
   ```

4. You now have to send this Certificate Signing Request (CSR) to a Certifying Authority (CA) to be signed. Once the CSR has been signed, you will have a real Certificate, which can be used by Apache. You can have a CSR signed by a commercial CA, or you can create your own CA to sign it. Commercial CAs usually ask you to post the CSR

into a web form, pay for the signing, and then send a signed Certificate, which you can store in a server.crt file. For more information about commercial CAs see the following locations:

i. Verisign http://digitalid.verisign.com/server/apacheNotice.htm
ii. Thawte http://www.thawte.com/
iii. CertiSign Certificadora Digital Ltda. http://www.certisign.com.br
iv. IKS GmbH http://www.iks-jena.de/leistungen/ca/
v. Uptime Commerce Ltd. http://www.uptimecommerce.com
vi. BelSign NV/SA http://www.belsign.beFor details on how to create your own CA, and use this to sign a CSR, see below. Once your CSR has been signed, you can see the details of the Certificate as follows:

`**$ openssl x509 -noout -text -in server.crt**`

5. You should now have two files: `server.key` 和 `server.crt` . These can be used as follows in your `httpd.conf` file:

```
        SSLCertificateFile    /path/to/this/server.crt
        SSLCertificateKeyFile /path/to/this/server.key
```

The `server.csr` file is no longer needed.

How do I create and use my own Certificate Authority (CA)?

The short answer is to use the `CA.sh` 或 `CA.pl` script provided by OpenSSL. Unless you have a good reason not to, you should use these for preference. If you cannot, you can create a self-signed Certificate as follows:

1. Create a RSA private key for your server (will be Triple-DES encrypted and PEM formatted):

   `**$ openssl genrsa -des3 -out server.key 1024**`

   Please backup this `host.key` file and the pass-phrase you entered in a secure location. You can see the details of this RSA private key by using the command:
   `**$ openssl rsa -noout -text -in server.key**`

   If necessary, you can also create a decrypted PEM version (not recommended) of this RSA private key with:

   `**$ openssl rsa -in server.key -out server.key.unsecure**`

2. Create a self-signed Certificate (X509 structure) with the RSA key you just created (output will be PEM formatted):

   `**$ openssl req -new -x509 -nodes -sha1 -days 365 -key server.key -out server.crt**`

This signs the server CSR and results in a `server.crt` file. You can see the details of this Certificate using:

```
**$ openssl x509 -noout -text -in server.crt**
```

How can I change the pass-phrase on my private key file?

You simply have to read it with the old pass-phrase and write it again, specifying the new pass-phrase. You can accomplish this with the following commands:

```
**$ openssl rsa -des3 -in server.key -out server.key.new**
**$ mv server.key.new server.key**
```

The first time you're asked for a PEM pass-phrase, you should enter the old pass-phrase. After that, you'll be asked again to enter a pass-phrase - this time, use the new pass-phrase. If you are asked to verify the pass-phrase, you'll need to enter the new pass-phrase a second time.

How can I get rid of the pass-phrase dialog at Apache startup time?

The reason this dialog pops up at startup and every re-start is that the RSA private key inside your server.key file is stored in encrypted format for security reasons. The pass-phrase is needed decrypt this file, so it can be read and parsed. Removing the pass-phrase removes a layer of security from your server - proceed with caution!

1. Remove the encryption from the RSA private key (while keeping a backup copy of the original file):

   ```
   **$ cp server.key server.key.org**
   **$ openssl rsa -in server.key.org -out server.key**
   ```

2. Make sure the server.key file is only readable by root:

   ```
   **$ chmod 400 server.key**
   ```

Now `server.key` contains an unencrypted copy of the key. If you point your server at this file, it will not prompt you for a pass-phrase. HOWEVER, if anyone gets this key they will be able to impersonate you on the net. PLEASE make sure that the permissions on this file are such that only root or the web server user can read it (preferably get your web server to start as root but run as another user, and have the key readable only by root).

As an alternative approach you can use the " `SSLPassPhraseDialog exec:/path/to/program` " facility. Bear in mind that this is neither more nor less secure, of course.

How do I verify that a private key matches its Certificate?

A private key contains a series of numbers. Two of these numbers form the "public key", the others are part of the "private key". The "public key" bits are included when you generate a CSR, and subsequently form part of the associated Certificate.

To check that the public key in your Certificate matches the public portion of your private key, you simply need to compare these numbers. To view the Certificate and the key run the commands:

```
**$ openssl x509 -noout -text -in server.crt**
**$ openssl rsa -noout -text -in server.key**
```

The 'modulus' and the 'public exponent' portions in the key and the Certificate must match. As the public exponent is usually 65537 and it's difficult to visually check that the long modulus numbers are the same, you can use the following approach:

```
**$ openssl x509 -noout -modulus -in server.crt | openssl md5**
**$ openssl rsa -noout -modulus -in server.key | openssl md5**
```

This leaves you with two rather shorter numbers to compare. It is, in theory, possible that these numbers may be the same, without the modulus numbers being the same, but the chances of this are overwhelmingly remote.

Should you wish to check to which key or certificate a particular CSR belongs you can perform the same calculation on the CSR as follows:

```
**$ openssl req -noout -modulus -in server.csr | openssl md5**
```

Why do connections fail with an "alert bad certificate" error?

Errors such as `OpenSSL: error:14094412: SSL routines:SSL3_READ_BYTES:sslv3 alert bad certificate` in the SSL logfile, are usually caused a browser which is unable to handle the server certificate/private-key. For example, Netscape Navigator 3.x is unable to handle RSA key lengths not equal to 1024 bits.

Why does my 2048-bit private key not work?

The private key sizes for SSL must be either 512 or 1024 bits, for compatibility with certain web browsers. A keysize of 1024 bits is recommended because keys larger than 1024 bits are incompatible with some versions of Netscape Navigator and Microsoft Internet Explorer, and with other browsers that use RSA's BSAFE cryptography toolkit.

Why is client authentication broken after upgrading from SSLeay version 0.8 to 0.9?

The CA certificates under the path you configured with `SSLCACertificatePath` are found by SSLeay through hash symlinks. These hash values are generated by the '`openssl x509 -noout -hash`' command. However, the algorithm used to calculate the hash for a certificate changed between SSLeay 0.8 and 0.9. You will need to remove all old hash symlinks and create new ones after upgrading. Use the `Makefile` provided by `mod_ssl`.

How can I convert a certificate from PEM to DER format?

The default certificate format for SSLeay/OpenSSL is PEM, which is simply Base64 encoded DER, with header and footer lines. For some applications (e.g. Microsoft Internet Explorer) you need the certificate in plain DER format. You can convert a PEM file `cert.pem` into the corresponding DER file `cert.der` using the following command:
`**$ openssl x509 -in cert.pem -out cert.der -outform DER**`

Why can't I find the `getca` 或 `getverisign` programs mentioned by Verisign, for installing my Verisign certificate?

Verisign has never provided specific instructions for Apache+mod_ssl. The instructions provided are for C2Net's Stronghold (a commercial Apache based server with SSL support).

To install your certificate, all you need to do is to save the certificate to a file, and give the name of that file to the `SSLCertificateFile` directive. You will also need to give it the key file. For more information, see the `SSLCertificateKeyFile` directive.

Can I use the Server Gated Cryptography (SGC) facility (aka Verisign Global ID) with mod_ssl?

Yes. `mod_ssl` has included support for the SGC facility since version 2.1. No special configuration is required - just use the Global ID as your server certificate. The *step up* of the clients is then automatically handled by `mod_ssl` at run-time.

Why do browsers complain that they cannot verify my Verisign Global ID server certificate?

Verisign uses an intermediate CA certificate between the root CA certificate (which is installed in the browsers) and the server certificate (which you installed on the server). You should have received this additional CA certificate from Verisign. If not, complain to them. Then, configure this certificate with the `SSLCertificateChainFile` directive. This ensures that the intermediate CA certificate is sent to the browser, filling the gap in the certificate chain.

The SSL Protocol

Why do I get lots of random SSL protocol errors under heavy server load?

There can be a number of reasons for this, but the main one is problems with the SSL session Cache specified by the `SSLSessionCache` directive. The DBM session cache is the most likely source of the problem, so using the SHM session cache (or no cache at all) may help.

Why does my webserver have a higher load, now that it serves SSL encrypted traffic?

SSL uses strong cryptographic encryption, which necessitates a lot of number crunching. When you request a webpage via HTTPS, everything (even the images) is encrypted before it is transferred. So increased HTTPS traffic leads to load increases.

Why do HTTPS connections to my server sometimes take up to 30 seconds to establish a connection?

This is usually caused by a `/dev/random` device for `SSLRandomSeed` which blocks the read(2) call until enough entropy is available to service the request. More information is available in the reference manual for the `SSLRandomSeed` directive.

What SSL Ciphers are supported by mod_ssl?

Usually, any SSL ciphers supported by the version of OpenSSL in use, are also supported by `mod_ssl`. Which ciphers are available can depend on the way you built OpenSSL. Typically, at least the following ciphers are supported:

1. RC4 with MD5
2. RC4 with MD5 (export version restricted to 40-bit key)
3. RC2 with MD5
4. RC2 with MD5 (export version restricted to 40-bit key)
5. IDEA with MD5
6. DES with MD5
7. Triple-DES with MD5

To determine the actual list of ciphers available, you should run the following:

```
$ openssl ciphers -v
```

Why do I get "no shared cipher" errors, when trying to use Anonymous Diffie-Hellman (ADH) ciphers?

By default, OpenSSL does *not* allow ADH ciphers, for security reasons. Please be sure you are aware of the potential side-effects if you choose to enable these ciphers.

In order to use Anonymous Diffie-Hellman (ADH) ciphers, you must build OpenSSL with "`-DSSL_ALLOW_ADH`", and then add "`ADH`" into your `SSLCipherSuite`.

Why do I get a 'no shared ciphers' error when connecting to my newly installed server?

Either you have made a mistake with your `SSLCipherSuite` directive (compare it with the pre-configured example in `httpd.conf-dist`) or you chose to use DSA/DH algorithms instead of RSA when you generated your private key and ignored or overlooked the warnings. If you have chosen DSA/DH, then your server cannot communicate using RSA-based SSL ciphers (at least until you configure an additional RSA-based certificate/key pair). Modern browsers like NS or IE can only communicate over SSL using RSA ciphers. The result is the "no shared ciphers" error. To fix this, regenerate your server certificate/key pair, using the RSA algorithm.

Why can't I use SSL with name-based/non-IP-based virtual hosts?

The reason is very technical, and a somewhat "chicken and egg" problem. The SSL protocol layer stays below the HTTP protocol layer and encapsulates HTTP. When an SSL connection (HTTPS) is established Apache/mod_ssl has to negotiate the SSL protocol parameters with the client. For this, mod_ssl has to consult the configuration of the virtual server (for instance it has to look for the cipher suite, the server certificate, etc.). But in order to go to the correct virtual server Apache has to know the `Host` HTTP header field. To do this, the HTTP request header has to be read. This cannot be done before the SSL handshake is finished, but the information is needed in order to complete the SSL handshake phase. Bingo!

Why is it not possible to use Name-Based Virtual Hosting to identify different SSL virtual hosts?

Name-Based Virtual Hosting is a very popular method of identifying different virtual hosts. It allows you to use the same IP address and the same port number for many different sites. When people move on to SSL, it seems natural to assume that the same method can be used to have lots of different SSL virtual hosts on the same server.

It comes as rather a shock to learn that it is impossible.

The reason is that the SSL protocol is a separate layer which encapsulates the HTTP protocol. So the SSL session is a separate transaction, that takes place before the HTTP session has begun. The server receives an SSL request on IP address X and port Y (usually 443). Since the SSL request does not contain any Host: field, the server has no way to decide which SSL virtual host to use. Usually, it will just use the first one it finds, which matches the port and IP address specified.

You can, of course, use Name-Based Virtual Hosting to identify many non-SSL virtual hosts (all on port 80, for example) and then have a single SSL virtual host (on port 443). But if you do this, you must make sure to put the non-SSL port number on the NameVirtualHost directive, e.g.

```
NameVirtualHost 192.168.1.1:80
```

Other workaround solutions include:

Using separate IP addresses for different SSL hosts. Using different port numbers for different SSL hosts.

How do I get SSL compression working?

Although SSL compression negotiation was defined in the specification of SSLv2 and TLS, it took until May 2004 for RFC 3749 to define DEFLATE as a negotiable standard compression method.

OpenSSL 0.9.8 started to support this by default when compiled with the `zlib` option. If both the client and the server support compression, it will be used. However, most clients still try to initially connect with an SSLv2 Hello. As SSLv2 did not include an array of prefered compression algorithms in its handshake, compression cannot be negotiated with these clients. If the client disables support for SSLv2, either an SSLv3 or TLS Hello may be sent, depending on which SSL library is used, and compression may be set up. You can verify whether clients make use of SSL compression by logging the `%{SSL_COMPRESS_METHOD}x` variable.

When I use Basic Authentication over HTTPS the lock icon in Netscape browsers stays unlocked when the dialog pops up. Does this mean the username/password is being sent unencrypted?

No, the username/password is transmitted encrypted. The icon in Netscape browsers is not actually synchronized with the SSL/TLS layer. It only toggles to the locked state when the first part of the actual webpage data is transferred, which may confuse people. The Basic Authentication facility is part of the HTTP layer, which is above the SSL/TLS layer in HTTPS. Before any HTTP data communication takes place in HTTPS, the SSL/TLS layer has already completed its handshake phase, and switched to encrypted communication. So don't be confused by this icon.

Why do I get I/O errors when connecting via HTTPS to an Apache+mod_ssl server with Microsoft Internet Explorer (MSIE)?

The first reason is that the SSL implementation in some MSIE versions has some subtle bugs related to the HTTP keep-alive facility and the SSL close notify alerts on socket connection close. Additionally the interaction between SSL and HTTP/1.1 features are problematic in some MSIE versions. You can work around these problems by forcing Apache not to use HTTP/1.1, keep-alive connections or send the SSL close notify messages to MSIE clients. This can be done by using the following directive in your SSL-aware virtual host section:

```
    SetEnvIf User-Agent ".*MSIE.*" \
             nokeepalive ssl-unclean-shutdown \
             downgrade-1.0 force-response-1.0
```

Further, some MSIE versions have problems with particular ciphers. Unfortunately, it is not possible to implement a MSIE-specific workaround for this, because the ciphers are needed as early as the SSL handshake phase. So a MSIE-specific `SetEnvIf` won't solve these problems. Instead, you will have to make more drastic adjustments to the global parameters. Before you decide to do this, make sure your clients really have problems. If not, do not make these changes - they will affect *all* your clients, MSIE or otherwise.

The next problem is that 56bit export versions of MSIE 5.x browsers have a broken SSLv3 implementation, which interacts badly with OpenSSL versions greater than 0.9.4. You can accept this and require your clients to upgrade their browsers, you can downgrade to OpenSSL 0.9.4 (not advised), or you can work around this, accepting that your workaround will affect other browsers too:

```
SSLProtocol all -SSLv3
```

will completely disables the SSLv3 protocol and allow those browsers to work. A better workaround is to disable only those ciphers which cause trouble.

```
SSLCipherSuite
    ALL:!ADH:**!EXPORT56**:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
```

This also allows the broken MSIE versions to work, but only removes the newer 56bit TLS ciphers.

Another problem with MSIE 5.x clients is that they refuse to connect to URLs of the form `https://12.34.56.78/` (where IP-addresses are used instead of the hostname), if the server is using the Server Gated Cryptography (SGC) facility. This can only be avoided by using the fully qualified domain name (FQDN) of the website in hyperlinks instead, because MSIE 5.x has an error in the way it handles the SGC negotiation.

And finally there are versions of MSIE which seem to require that an SSL session can be reused (a totally non standard-conforming behaviour, of course). Connecting with those MSIE versions only work if a SSL session cache is used. So, as a work-around, make sure you are using a session cache (see the `SSLSessionCache` directive).

Why do I get I/O errors, or the message "Netscape has encountered bad data from the server", when connecting via HTTPS to an Apache+mod_ssl server with Netscape Navigator?

This usually occurs when you have created a new server certificate for a given domain, but had previously told your browser to always accept the old server certificate. Once you clear the entry for the old certificate from your browser, everything should be fine. Netscape's SSL implementation is correct, so when you encounter I/O errors with Netscape Navigator it is usually caused by the configured certificates.

mod_ssl Support

What information resources are available in case of mod_ssl problems?

The following information resources are available. In case of problems you should search here first.

Answers in the User Manual's F.A.Q. List (this)

http://httpd.apache.org/docs/2.2/ssl/ssl_faq.html First check the F.A.Q. (this text). If your problem is a common one, it may have been answered several times before, and been included in this doc.

Postings from the modssl-users Support Mailing List http://www.modssl.org/support/

Search for your problem in the archives of the modssl-users mailing list. You're probably not the first person to have had this problem!

What support contacts are available in case of mod_ssl problems?

The following lists all support possibilities for mod*ssl, in order of preference. Please go through these possibilities _in this order* - don't just pick the one you like the look of.

1. *Send a Problem Report to the modssl-users Support Mailing List* modssl-users@modssl.org This is the preferred way of submitting your problem report, because this way, others can see the problem, and learn from any answers. You must subscribe to the list first, but you can then easily discuss your problem with both the author and the whole mod_ssl user community.
2. *Send a Problem Report to the Apache httpd Users Support Mailing List* users@httpd.apache.org This is the second way of submitting your problem report. Again, you must subscribe to the list first, but you can then easily discuss your problem with the whole Apache httpd user community.
3. *Write a Problem Report in the Bug Database* http://httpd.apache.org/bug_report.html This is the last way of submitting your problem report. You should only do this if you've already posted to the mailing lists, and had no success. Please follow the instructions on the above page *carefully*.

What information should I provide when writing a bug report?

You should always provide at least the following information:

Apache and OpenSSL version information

The Apache version can be determined by running `httpd -v`. The OpenSSL version can be determined by running `openssl version`. Alternatively, if you have Lynx installed, you can run the command `lynx -mime_header http://localhost/ | grep Server` to gather this information in a single step.

The details on how you built and installed Apache+mod_ssl+OpenSSL

For this you can provide a logfile of your terminal session which shows the configuration and install steps. If this is not possible, you should at least provide the `configure` command line you used.

In case of core dumps please include a Backtrace

If your Apache+mod_ssl+OpenSSL dumps its core, please attach a stack-frame "backtrace" (see below for information on how to get this). Without this information, the reason for your core dump cannot be found

A detailed description of your problem

Don't laugh, we really mean it! Many problem reports don't include a description of what the actual problem is. Without this, it's very difficult for anyone to help you. So, it's in your own interest (you want the problem be solved, don't you?) to include as much detail as possible, please. Of course, you should still include all the essentials above too.

I had a core dump, can you help me?

In general no, at least not unless you provide more details about the code location where Apache dumped core. What is usually always required in order to help you is a backtrace (see next question). Without this information it is mostly impossible to find the problem and help you in fixing it.

How do I get a backtrace, to help find the reason for my core dump?

Following are the steps you will need to complete, to get a backtrace:

1. Make sure you have debugging symbols available, at least in Apache. On platforms where you use GCC/GDB, you will have to build Apache+mod_ssl with " `OPTIM="-g -ggdb3"` " to get this. On other platforms at least " `OPTIM="-g"` " is needed.
2. Start the server and try to reproduce the core-dump. For this you may want to use a directive like " `CoreDumpDirectory /tmp` " to make sure that the core-dump file can be written. This should result in a `/tmp/core` 或 `/tmp/httpd.core` file. If you don't get one of these, try running your server under a non-root UID. Many modern kernels do not allow a process to dump core after it has done a `setuid()` (unless it does an `exec()`) for security reasons (there can be privileged information left over in memory). If necessary, you can run `/path/to/httpd -X` manually to force Apache to not fork.
3. Analyze the core-dump. For this, run `gdb /path/to/httpd /tmp/httpd.core` or a similar command. In GDB, all you have to do then is to enter `bt` , and voila, you get the backtrace. For other debuggers consult your local debugger manual.

如何.../指南

如何.../指南

认证

认证(Authentication)是指任何识别用户身份的过程。授权(Authorization)是允许特定用户访问特定区域或信息的过程。

参见：认证、授权、访问控制

CGI动态页面

CGI(公共网关接口)定义了web服务器与外部内容生成程序之间交互的方法，通常是指CGI程序或者CGI脚本，它是在网站上实现动态页面的最简单和常用的方法。本文将对如何在Apache web服务器上建立CGI以及如何编写CGI程序进行介绍。

参见：CGI动态页面

`.htaccess` 文件

`.htaccess` 文件(或者"分布式配置文件")提供了针对每个目录改变配置的方法，即在一个特定的目录中放置一个包含指令的文件，其中的指令作用于此目录及其所有子目录。

See: `.htaccess` 文件

服务器端包含

SSI是嵌入HTML页面中的指令，在页面被提供时由服务器进行运算，以对现有HTML页面增加动态生成的内容，而无须通过CGI程序提供其整个页面，或者使用其他动态技术。

See: 服务器端包含(SSI)

用户网站目录

在多用户系统中，用 `UserDir` 指令可以允许每个用户在其宿主目录中拥有一个网络站点。使用URL `http://example.com/~username/` 的访问者可以获得用户" `username` "的宿主目录中的内容或者用 `UserDir` 指定的子目录中的内容。

See: 用户网站目录(`public_html`)

认证、授权、访问控制

认证(Authentication)是指任何识别用户身份的过程。授权(Authorization)是允许特定用户访问特定区域或信息的过程。

相关模块和指令

认证和授权涉及到三组模块。通常，你需要从每一组中选择至少一个模块。

- 认证类型模块(参见 `AuthType` 指令)
    - `mod_auth_basic`
    - `mod_auth_digest`
- 认证支持模块
    - `mod_authn_alias`
    - `mod_authn_anon`
    - `mod_authn_dbd`
    - `mod_authn_dbm`
    - `mod_authn_default`
    - `mod_authn_file`
    - `mod_authnz_ldap`
- 授权支持模块(参见 `Require` 指令)
    - `mod_authnz_ldap`
    - `mod_authz_dbm`
    - `mod_authz_default`
    - `mod_authz_groupfile`
    - `mod_authz_owner`
    - `mod_authz_user`

`mod_authnz_ldap` 模块既包含认证功能也包含授权功能。 `mod_authn_alias` 模块自身并不实现认证功能，但是允许其它认证支持模块以更灵活的方式进行配置。

`mod_authz_host` 模块提供基于主机名、IP地址、请求特征的访问控制，但并不属于认证支持系统。

简介

如果网站上有些敏感信息或只希望为一个小群体所访问，本文阐述的方法能确保用户只能访问被允许的资源。

本文涵盖了保护站点资源的"标准"方法，大多数管理员将要用到这些方法。

先决条件

本文中讨论的指令应该被放进主配置文件(通常在 `<Directory>` 段中)或者针对单个目录的配置文件(`.htaccess` 文件)中。

如果你打算使用 `.htaccess` 文件，则必须设置服务器以允许在这些文件中使用认证指令，即用 `AllowOverride` 指令指定哪些指令在针对单个目录的配置文件中有效。

既然本文讨论认证，就应该对 `AllowOverride` 这样设置：

```
AllowOverride AuthConfig
```

如果你希望把这些指令直接写入主配置文件，当然就需要具有对主配置文件的写权限。

而且，你需要对服务器的目录结构有所了解，以确定某些文件的位置。这个并不难，需要时我们会做适当的说明。

启用认证

先介绍用密码来保护服务器上的目录。

首先需要建立一个密码文件。这个文件应该放在不能被网络访问的位置，以避免被下载。例如，如果 `/usr/local/apache/htdocs` 以外的空间不能被网络访问，那么可以考虑把密码文件放在 `/usr/local/apache/passwd` 目录中。

Apache在其安装目录的 `bin` 子目录中提供了 `htpasswd` 工具，用于建立密码文件，可以这样使用：

```
htpasswd -c /usr/local/apache/passwd/passwords rbowen
```

`htpasswd` 会要你输入密码，并要求重新输入以进行确认：

```
# htpasswd -c /usr/local/apache/passwd/passwords rbowen

New password: mypassword

Re-type new password: mypassword

Adding password for user rbowen
```

如果 `htpasswd` 不在搜索路径中，则必须使用完整路径，如： `/usr/local/apache/bin/htpasswd`

然后修改 `httpd.conf` 或 `.htaccess` 文件，指示服务器允许哪些用户访问并向用户索取密码。若要保护 `/usr/local/apache/htdocs/secret` 目录，则可以将下列指令写入 `/usr/local/apache/htdocs/secret/.htaccess` 或者 `httpd.conf` 的<Directory /usr/local/apache/apache/htdocs/secret>段。

```
AuthType Basic
AuthName "Restricted Files"
AuthUserFile /usr/local/apache/passwd/passwords
Require user rbowen
```

让我们逐个解释这些指令。 `AuthType` 指令选择对用户实施认证的方法，最常用的是由 `mod_auth_basic` 提供的 `Basic` 。必须认识到的很重要的一点是，Basic认证方法并不加密来自用户浏览器的密码，因此，不应该用于高度敏感的数据。Apache中还有另一种更安全的认证方法" `AuthType Digest` "，即由 `mod_auth_digest` 供的摘要认证。目前，只有最新的浏览器版本才支持摘要认证。

`AuthName` 指令设置了使用认证的<dfn class="calibre27">域(Realm)</dfn>，它起两个作用，首先，此域会出现在显示给用户的密码提问对话框中，其次，也帮助客户端程序确定应该发送哪个密码。

所以，如果一个用户已经在 `"Restricted Files"` 域通过了认证，则客户端就可以尝试使用同样的密码来访问同一个服务器上任何名为 `"Restricted Files"` 域的其他部分，从而使多个受限区域使用同一个密码，以避免用户重复输入。当然，出于安全考虑，如果服务器变了，客户端始终会要求重新输入密码。

`AuthUserFile` 指令设置了密码文件的位置，也就是刚才我们用 `htpasswd` 建立的文件。如果用户很多则认证速度会很慢，因为对每个请求都必须搜索这个纯文本文件，对此，Apache还支持把用户信息存入快速的数据库文件， `mod_authn_dbm` 模块提供了 `AuthDBMUserFile` 指令，并可以用 `dbmmanage` 程序建立和操作这些数据库。Apache模块数据库中还提供了许多其他第三方模块提供的认证选项。

最后， `Require` 指令设置了允许访问受保护区域的用户，下一节将对 `Require` 指令作详细说明。

允许多人访问

上述指令只允许一个人(一个叫 `rbowen` 的用户)访问这个目录，但是多数情况下，都需要允许多人访问，所以就要用到 `AuthGroupFile` 指令。

如果想允许多人访问，那么就必须建立一个组文件以确定组中的用户。其格式很简单，可以用你喜欢的编辑器建立，例如：

```
GroupName: rbowen dpitts sungo rshersey
```

它只是每组一行的一个用空格分隔的组成员列表。

向已有的密码文件中增加一个用户，可以输入：

```
htpasswd /usr/local/apache/passwd/passwords dpitts
```

程序的提示和上面的一样，但是它会追加到已有的文件中，而不是建一个新文件(参数 `-c` 可以强制建立新的密码文件)。

现在，需要将 `.htaccess` 文件修改成这样：

```
AuthType Basic
AuthName "By Invitation Only"
AuthUserFile /usr/local/apache/passwd/passwords
AuthGroupFile /usr/local/apache/passwd/groups
Require group GroupName
```

现在， `GroupName` 组中的成员都在 `password` 文件中有一个相应的记录，从而允许他们输入正确的密码进行访问。

除了建立组文件，还有另一种途径允许多人访问，就是使用如下指令：

```
Require valid-user
```

使用上述指令，而不是 `Require user rbowen` ，可以允许密码文件中的所有用户使用正确的密码进行访问。通过为每个组建立一个密码文件，这里甚至允许列举各个组，其优点是Apache只需要检查一个文件(而不是两个)，其缺点是，必须维护众多密码文件，而且要确保 `AuthUserFile` 指定了一个正确的密码文件。

可能存在的问题

由于采用了Basic认证的方法，每次向服务器请求甚至刷新一个受保护的页面或图片时都必须校验用户名和密码，为此，必须打开密码文件并逐行搜索用户名，因此，服务器响应速度会受一些影响，受影响的程度与密码文件的大小成正比。

所以，对密码文件中的用户总数存在一个实际上的上限，此上限取决于特定的服务器机器的性能，但是一般有几百个用户就会对响应速度有非常明显的影响，在这种情况下，可以考虑用其他认证方法。

其他认证方法

基于用户名和密码的认证只是方法之一，时常会有不需要知道来访者是谁，只需要知道来自哪里的情况。

`Allow` 和 `Deny` 指令可以允许或拒绝来自特定主机名或主机地址的访问，同时，`Order` 指令告诉Apache处理这两个指令的顺序，以改变过滤器。

这些指令的用法：

```
Allow from <var class="calibre40">address</var>
```

address可以是一个IP地址(或者IP地址的一部分)，也可以是一个完整的域名(或者域名的一部分)，还可以同时指定多个IP地址和域名。

比如，要拒绝不受欢迎的兜售垃圾的站点：

```
Deny from 205.252.46.165
```

这样，这个指令所管辖的区域将拒绝所有来自该地址的访问。除了指定IP地址，也可以指定域名，如：

```
Deny from <var class="calibre40">host.example.com</var>
```

另外，还可以指定地址或域名的一部分来阻止一个群体：

```
Deny from <var class="calibre40">192.101.205</var>
Deny from <var class="calibre40">cyberthugs.com</var> <var class="calibre40">moreid
Deny from ke
```

`Order` 可以组合 `Deny` 和 `Allow` 指令，以保证在允许一个群体访问的同时，对其中的一些又加以限制：

```
Order deny,allow
Deny from all
Allow from <var class="calibre40">dev.example.com</var>
```

只列出 `Allow` 指令不会得到你想要的结果，因为它在允许指定对象访问的同时并不禁止其他未列出的对象的访问。所以上例使用的方法是：首先拒绝任何人，然后允许来自特定主机的访问。

更多信息

`mod_auth_basic`和`mod_authz_host`文档中有更多的有关资料。`mod_authn_alias`同样有助于简化认证配置。

CGI动态页面

简介

相关模块

- `mod_alias`
- `mod_cgi`

相关指令

- `AddHandler`
- `Options`
- `ScriptAlias`

CGI(公共网关接口)定义了web服务器与外部内容生成程序之间交互的方法，通常是指CGI程序或者CGI脚本，它是在网站上实现动态页面的最简单和常用的方法。本文将对如何在Apache web服务器上建立CGI以及如何编写CGI程序进行介绍。

配置Apache以允许CGI

要让CGI程序能正常运作，必须配置Apache以允许CGI的执行，其方法有多种。

ScriptAlias

`ScriptAlias` 指令使Apache允许执行一个特定目录中的CGI程序。当客户端请求此特定目录中的资源时，Apache假定其中所有的文件都是CGI程序并试图运行它。

`ScriptAlias` 指令形如：

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
```

如果Apache被安装到默认位置，默认的配置文件 `httpd.conf` 中就会有上述配置。 `ScriptAlias` 与 `Alias` 指令非常相似，都是定义了映射到一个特定目录的URL前缀，两者一般都用于指定位于 `DocumentRoot` 以外的目录，其不同之处是 `ScriptAlias` 又多了一层含义，即URL前缀后面的任何文件都被视为CGI程序。所以，上述例子会指示Apache：任何以 `/cgi-bin/` 开头的资源都将映射到 `/usr/local/apache2/cgi-bin/` 目录中，且视之为CGI程序。

例如，如果有URL为 `http://www.example.com/cgi-bin/test.pl` 的请求，Apache会试图执行 `/usr/local/apache2/cgi-bin/test.pl` 文件并返回其输出。当然，这个文件必须存在而且可执行，并以特定的方法产生输出，否则Apache返回一个出错消息。

ScriptAlias目录以外的CGI

由于安全原因，CGI程序通常被限制在 `ScriptAlias` 指定的目录中，这样，管理员就可以严格控制谁可以使用CGI程序。但是，如果采取了恰当的安全措施，则没有理由不允许其他目录中的CGI程序运行。比如，你可能希望用户在 `UserDir` 指定的宿主目录中存放页面，而他们有自己的CGI程序，但无权访问 `cgi-bin` 目录，这样，就产生了运行其他目录中CGI程序的需求。

允许CGI在任意目录执行需要两个步骤：第一步，必须用 `AddHandler` 或 `SetHandler` 指令激活 `cgi-script` 处理器。第二步，必须在 `Options` 指令中启用 `ExecCGI` 选项。

用Options显式地允许CGI的执行

可以在主配置文件中，使用 `Options` 指令显式地允许特定目录中CGI的执行：

```
        <Directory /usr/local/apache2/htdocs/somedir>

Options +ExecCGI
        </Directory>
```

上述指令使Apache允许CGI文件的执行。另外，还必须告诉服务器哪些文件是CGI文件。下面的 `AddHandler` 指令告诉服务器所有带有 `cgi` 或 `pl` 后缀的文件是CGI程序：

```
        AddHandler cgi-script .cgi .pl
```

.htaccess文件

`.htaccess` 指南示范了怎样在没有权限修改 `httpd.conf` 文件的情况下激活CGI程序。

用户目录

为了允许用户目录中所有以" `.cgi` "结尾的文件作为CGI程序执行，你可以使用以下配置：

```
      <Directory /home/*/public_html>

Options +ExecCGI

        AddHandler cgi-script .cgi
      </Directory>
```

如果你想在用户目录中指定一个 `cgi-bin` 子目录，其中所有的文件都被当作CGI程序，你可以这样配置：

```
      <Directory /home/*/public_html/cgi-bin>

  Options ExecCGI

        SetHandler cgi-script
      </Directory>
```

编写CGI程序

编写CGI程序和"常规"程序之间有两个主要的不同。

首先，在CGI程序的所有输出前面必须有一个HTTP的MIME类型的头，对客户端指明所接收内容的类型，大多数情况下，像这样：

```
      Content-type: text/html
```

其次，输出要求是HTML形式的，或者是浏览器可以显示的其他某种形式。多数情况下，输出是HTML形式的，但偶然也会输出一个gif图片或者其他非HTML的内容。

除了这两点，编写CGI程序和编写其他程序大致相同。

第一个CGI程序

这个CGI程序的例子在浏览器中打印一行文字。把下列存为 `first.pl` 文件，并放在你的 `cgi-bin` 目录中。

```
        #!/usr/bin/perl

        print "Content-type: text/html\n\n";

        print "Hello, World.";
```

即使不熟悉Perl语言，你也应该能看出它干了什么。第一行，告诉Apache这个文件可以用 `/usr/bin/perl` (或者任何你正在使用的shell)解释并执行。第二行，打印上述要求的内容类型说明，并带有两个换行，在头后面留出空行，以示HTTP头的结束。第三行，打印文字"Hello, World."。程序到此结束。

打开你喜欢的浏览器并输入地址：

```
        http://www.example.com/cgi-bin/first.pl
```

或者是你存放程序的其他位置，就可以在浏览器窗口中看到一行：`Hello, World.` 。虽然并不怎么激动人心，但是一旦这个程序能正常运行，那么就可能运行其他任何程序。

程序还是不能运行！

使用浏览器从网络访问CGI程序，可能会发生四种情况：

CGI程序的输出

太好了！这说明一切正常。如果输出正常但是浏览器处理出错，请确认你在CGI程序中使用了正确的 `Content-Type` 。

CGI程序的源代码或者一个"POST Method Not Allowed"消息

这说明Apache没有被正确配置以执行CGI程序，重新阅读配置Apache看看遗漏了什么。

一个以"Forbidden"开头的消息

这说明有权限问题。参考Apache错误日志和下面的文件权限。

一个"Internal Server Error"消息

查阅Apache错误日志，可以找到CGI程序产生的出错消息"Premature end of script headers"。对此，需要检查下列各项，以找出不能产生正确HTTP头的原因。

文件的权限

记住，服务器不是以你的用户身份运行的，在服务器启动后，拥有的是一个非特权用户的权限(通常是 `nobody` 或 `www`)而需要更大的权限以允许文件的执行。通常，给予 `nobody` 足够的权限以执行文件的方法是，对文件赋予任何人皆可执行的权限：

```
chmod a+x first.pl
```

另外，如果需要对其他文件进行读取或写入，也必须对这些文件赋予正确的权限。

路径信息和环境变量

当你在命令行执行一个程序，某些信息会自动传给shell而无须你操心，比如 `PATH` ，告诉shell你所引用的文件可以在哪儿找到。

但是，在CGI程序通过web服务器执行时，则没有此 `PATH` ，所以，你在CGI程序中引用的任何程序(如 `sendmail`)都必须指定其完整的路径，使shell能找到它们以执行你的CGI程序。

一种普通的用法是，在CGI程序的第一行中指明解释器(通常是 `perl`)，形如：

```
#!/usr/bin/perl
```

必须保证它的确指向解释器。

另外，如果CGI程序依赖于某些环境变量，你要确保所需要的变量已经正确的由Apache进行了传递。

程序错误

多数CGI程序失败的原因在于程序本身有问题，尤其是在已经消除上述两种错误而CGI挂起的情况下。在用浏览器测试以前，先在命令行中执行你的程序，能够发现大多数的问题。比如：

```
cd /usr/local/apache2/cgi-bin

./first.pl
```

(不要调用 `perl` 解释程序，因为shell和Apache会根据脚本第一行的路径信息找到解释器)

你最先看到的输出内容应当是一组HTTP头，包括 `Content-Type` 和结尾的空行。如果你看到了别的什么东西，那么当你在服务器上试运行时，Apache会返回 `Premature end of script headers` 错误。参见上面的编写CGI程序以获得更多信息。

错误日志

错误日志是你的朋友。任何错误都会在错误日志中有所记载，所以你应该首先查看它。如果你的网站空间提供者不允许访问错误日志，那么你应该考虑换一个空间提供者。学会阅读错误日志，可以快速找出问题并快速解决。

Suexec

suexec允许CGI程序根据其所在虚拟主机或用户宿主目录的不同而以不同的用户权限运行。suexec有极其严格的权限校验，任何校验失败都会使CGI程序遭遇 `Premature end of script headers` 错误。

为了检查你是否使用了suexec，运行 `apachectl -V` 并检查 `SUEXEC_BIN` 的位置。如果Apache在启动时发现 `suexec` 二进制文件正存在于此，那么suexec将会被激活。

除非你很精通suexec，否则请不要使用它。要禁用它，只要删除(或重命名) `SUEXEC_BIN` 所指定位置的 `suexec` 二进制文件并重启服务器就可以了。如果你又想启用它，请首先阅读suexec文档以详细了解其运行机制，然后运行 `suexec -V` 命令找到suexec日志文件，并使用该日志文件找到你违反了哪条判断规则。

幕后是怎样操作的?

当你的CGI编程逐渐深入，理解幕后的操作(尤其是浏览器和服务器之间是如何通讯的)就变得很有用了。因为虽然成功地写了一个程序打印"Hello, World"，但并没有实际的用处。

环境变量

环境变量是使用计算机时到处都会用到的变量，比如路径(对实际文件的一个搜索路径以补全你的输入)、你的用户名以及你的终端类型等等。在命令行输入 `env` ，可以得到当天标准的环境变量列表。

在CGI处理过程中，服务器和浏览器都会设置环境变量，比如浏览器类型(Netscape、IE、Lynx)、服务器类型(Apache、IIS、WebSite)以及将要执行的CGI程序名称等等。

所有这些变量对CGI程序员都有效，但只是客户端-服务器通讯的一半内容。完整的变量列表参见http://hoohoo.ncsa.uiuc.edu/cgi/env.html

这个简单的CGI程序列出了所有的环境变量，Apache发行版的 `cgi-bin` 目录中还有一个类似的程序。注意，有些变量是必须的，有些则是可选的，所以你可能会看见一些官方列表中没有的变量。另外，Apache有多种方法可以在默认提供的变量之外增加你的专用环境变量。

```
        #!/usr/bin/perl

        print "Content-type: text/html\n\n";

        foreach $key (keys %ENV) {

print "$key --> $ENV{$key}<br>";
        }
```

STDIN 和 STDOUT

服务器和客户端之间的其他通讯都通过标准输入设备(`STDIN`)和标准输出设备(`STDOUT`)完成。通常， `STDIN` 是指键盘或者一个程序所作用的一个文件， `STDOUT` 指控制台或显示器。

当你 `POST` 一个网络表格到一个CGI程序时，表格中的数据被捆扎为一个特殊形式通过 `STDIN` 传送给CGI程序，这样，这个程序就可以处理这些数据，仿佛这些数据是来自键盘或者一个文件。

这种"特殊形式"很简单，一个字段名称及其值，中间用等号(=)连接，多个这样的字段对用与符号(&)连接。非常规字符，如空格、"&"号和"="号，被转换为其等值的十六进制以免出问题。整个字符串形如：

```
name=Rich%20Bowen&city=Lexington&state=KY&sidekick=Squirrel%20Monkey
```

有时，你会发现URL后面也会带有这样的字符串。这种形式会使服务器以这个字符串的内容设置环境变量 `QUERY_STRING` ，称为 `GET` 请求。你的HTML表格在 `FORM` 标签中设置 `METHOD` 属性，以指定传送数据的动作使用 `GET` 或 `POST` 。

你的程序必须把这个字符串分解以获得有用信息。所幸，有库和模块可以帮助你处理这些数据，还有为你的CGI程序达成其他目的的处理器。

CGI模块/库

编写CGI程序时，你应该考虑使用代码库或模块来完成大多数琐碎的工作，以减少错误并更快地开发。

如果用Perl语言编写CGI程序，可用的模块见CPAN，最常用的模块是 `CGI.pm` 。也可以考虑用 `CGI::Lite` ，它实现了一个在多数程序中所有必须的最小功能集。

如果用C语言编写CGI程序，则有很多选择，其中之一是 `CGIC` 库，来自 http://www.boutell.com/cgic/

更多信息

网上有大量的CGI资源。可以在Usenet组comp.infosystems.www.authoring.cgi和别人讨论CGI相关问题。HTML Writers Guild 的邮件列表是一个优秀的问题解答资源。更多资源在http://www.hwg.org/lists/hwg-servers/

另外，还可以阅读CGI规范，其中有CGI程序操作的所有细节，原始版本见NCSA，另有一个更新草案见Common Gateway Interface RFC project

当你向一个邮件列表或者新闻组提交CGI相关问题时，你应该确保提供了足够的信息以更容易地发现并解决问题，诸如：发生了什么事、你希望得到什么结果、结果与你所期望的有什么出入、你运行的服务器、CGI程序是用什么语言编写的、如果可能就提供那个讨厌的代码。

注意，不要把CGI相关问题提交到Apache bug数据库，除非你坚信发现的是Apache源代码中的问题。

服务器端包含入门

服务器端包含提供了一种对现有HTML文档增加动态内容的方法。

简介

相关模块

- `mod_include`
- `mod_cgi`
- `mod_expires`

相关指令

- `Options`
- `XBitHack`
- `AddType`
- `SetOutputFilter`
- `BrowserMatchNoCase`

本文针对服务器端包含(SSI)讨论如何配置服务器以允许SSI，并介绍一些对现有HTML页面增加动态内容的基本SSI技术。

本文后部将讨论用SSI做一些稍微高级的事情，比如SSI指令中的条件语句。

什么是SSI？

SSI是嵌入HTML页面中的指令，在页面被提供时由服务器进行运算，以对现有HTML页面增加动态生成的内容，而无须通过CGI程序提供其整个页面，或者使用其他动态技术。

至于什么时候应当用SSI，而什么时候应当用某些程序生成整个页面，取决于页面中有多少内容是静态的，又有多少内容需要在每次页面被提供时重新计算。SSI是一种增加小段动态信息的好方法，比如当前时间。如果你的页面大部分内容是在被提供时动态生成的，那就要另找方案了。

配置服务器以允许SSI

要使服务器允许SSI，必须在 `httpd.conf` 或 `.htaccess` 文件中有如下配置：

```
Options +Includes
```

这样就告诉服务器允许解析文件中的SSI指令。注意，在多数配置中，多个 `Options` 指令会互相覆盖，所以可能需要对使用SSI的目录专门使用一个 `Options` 指令，以确保其有效。

并非所有文件中的SSI指令都会被解析，必须告诉Apache应该解析哪些文件。有两种方法使Apache解析带有特定后缀名的文件，比如： `.shtml` ，配置如下：

```
        AddType text/html .shtml

    AddOutputFilter INCLUDES .shtml
```

该方法的缺点之一是，为了使文件具有 `.shtml` 后缀从而执行其中的指令，需要加入SSI指令的现有文件的名字，以及所有指向此页面的连接。

另一种方法是，使用 `XBitHack` 指令：

```
        XBitHack on
```

`XBitHack` 告诉Apache解析所有设置了执行位的文件中的SSI指令。这样，无需修改文件名，只要用 `chmod` 使文件变成可执行的，就可以对现有页面增加SSI指令。

```
        chmod +x pagename.html
```

这里简要说明一点：偶然会有人向你推荐，无须用带 `.shtml` 的文件名，只要使Apache解析所有 `.html` 文件的SSI就可以了。那些人可能没听说过 `XBitHack` 。要知道，这样做会使Apache在发送文件到客户端之前通读此文件，即使其中并没有任何SSI指令，从而对速度有很不利的影响，所以这并不是好办法。

当然，在Windows上，没有对应的执行位可以设置，所以对你的配置方法就有一些限制。

在默认配置的情况下，Apache不会为SSI页面发送最后修改日期或者内容长度的HTTP头，因为这些值对动态页面来说难以确定。这样会阻止页面被缓冲，导致客户端性能有明显的下降。有两种解决方法：

1. 设置 `XBitHack Full` ，告诉Apache在判断最后修改日期时，只查看被请求文件本身的日期，而忽略其中包含的其它文件的修改日期。
2. 使用 `mod_expires` 提供的指令为文件设置一个明确的过期时间，并告诉浏览器和代理这个文件可以被缓冲。

基本SSI指令

SSI指令有如下语法：

```
        <!--#element attribute=value attribute=value ... -->
```

类似于HTML注释，即使没有正确配置SSI，它也不会被浏览器显示，但在HTML代码中可见。而若正确配置了SSI，则指令会被其结果所取代。

其中的元素可以有许多，我们会在下一个版本的文档中讨论其中的大多数，而在这里，仅举几个SSI的例子。

今天的日期

```
<!--#echo var="DATE_LOCAL" -->
```

`echo` 元素用于显示一个变量的值。标准变量有很多，其中包含对CGI程序有效的所有环境变量。并且还可以用 `set` 元素定义你自己的专用变量。

如果你不喜欢这种日期格式，可以用 `config` 元素的 `timefmt` 属性，改变其格式。

```
<!--#config timefmt="%A %B %d, %Y" -->

Today is <!--#echo var="DATE_LOCAL" -->
```

文件的修改日期

```
This document last modified <!--#flastmod file="index.html" -->
```

这个元素使用 `timefmt` 的格式配置。

包含一个CGI程序的输出结果

这也是SSI很常见的一个用途：包含一个CGI程序的输出，比如人人喜欢的"点击计数器"。

```
<!--#include virtual="/cgi-bin/counter.pl" -->
```

附加的例子

以下是一些在HTML中使用SSI的特殊例子。

文档是什么时候被修改的?

前面我们提到过可以用SSI告诉用户文档是什么时候被修改的，但是具体实现方法却未说明。将以下代码放到HTML中，会在页面中产生一个时间戳，当然，你必须首先按前面的方法启用SSI 。

```
<!--#config timefmt="%A %B %d, %Y" -->
This file last modified <!--#flastmod file="ssi.shtml" -->
```

不用说，你应该用你实际引用的文件名来替换 `ssi.shtml` ，所以，如果你想简单地在所有文件中使用这段通用代码以达到这个目的，这个方法就并不方便，就需要用到 `LAST_MODIFIED` 变量：

```
<!--#config timefmt="%D" -->
This file last modified <!--#echo var="LAST_MODIFIED" -->
```

有关 `timefmt` 格式的细节，可以到google查找 `strftime` ，其语法是相同的。

包含一个标准页脚

当你管理一个拥有许多页面的站点，你会发现对所有页面同时做改动是很痛苦的，尤其是在试图对所有页面维持某种标准视觉效果的时候。

使用包含一个页眉/页脚的方法，可以减轻修改的负担。你只要制作一个页脚文件，并用 `include` 命令包含到每个页面即可。 `include` 元素能按 `file` 属性或 `virtual` 属性判断应该包含的文件。 `file` 属性是一个相对于当前目录的文件路径，即不能是一个绝对路径(以"/"开头)或包含"../"的路径。 `virtual` 属性可能更有用，它是一个相对于被提供的文档的URL，可以以"/"开头，但必须与被提供的文档位于同一服务器上。

```
<!--#include virtual="/footer.html" -->
```

SSI指令和页脚文件相结合使用是很有用的，比如在页脚文件中使用 `LAST_MODIFIED` 指令。SSI指令可以出现在包含文件中，而 `include` 可以嵌套，即一个包含文件还可以再包含另外一个。

我还能设置其它什么？

`config` 除了能设置时间格式，还有两种用途。

当SSI指令发生错误时，会产生如下消息：

```
[an error occurred while processing this directive]
```

为了改变消息的形式，可以使用 `config` 元素的 `errmsg` 属性：

```
<!--#config errmsg="[It appears that you don't know how to use SSI]" -->
```

希望最终用户永远也不会看到这个消息，因为在网站投入运行之前你已经把这些问题都解决了。是吗？

还可以使用 `config` 的 `sizefmt` 属性设置返回的文件大小的格式，或者是以 `bytes` 为单位，或者是以Kb或Mb为单位的 `简写(abbrev)` 。

执行命令

我期望未来几个月内能再写一篇小型的CGI程序使用SSI的文章，而这里仅介绍 `exec` 的使用。SSI确实可以利用shell(`/bin/sh` ，精确地说，还可以是Win32中的DOS shell)来执行命令。下例产生一个目录列表：

```
<pre>
<!--#exec cmd="ls" -->
</pre>
```

或者在Windows中：

```
<pre>
<!--#exec cmd="dir" -->
</pre>
```

你可能会发现，在Windows中这个指令的结果有些奇怪， `dir` 的输出中包含有字符串"< `dir` >"，它会使浏览器产生混淆。

注意，这个功能是极度危险的，因为它会执行任何包含在 `exec` 标记中的命令。如果用户有可能修改你的网页内容，比如"留言本"，那么你一定要关闭这个功能。可以在 `Options` 指令中加上 `IncludesNOEXEC` 参数，以关闭 `exec` 功能，同时又保留SSI。

高级SSI技术

除了分离内容，Apache SSI还有设置变量的操作，并且还可以将这些变量用在比较和条件表达式中。

警告

本文中讨论的大多数功能仅在Apache1.2及更新版本中有效。如果你运行的不是Apache1.2及更新版本，请立刻或者尽快升级，现在就动手，我们会等你弄好了再继续往下讲。

设置变量

使用 `set` 指令可以设置变量以备后用，其语法是：

```
<!--#set var="name" value="Rich" -->
```

除了设置字面变量以外，还可以设置其他任何变量，比如环境变量和此前提到过的一些变量(如 `LAST_MODIFIED`)，作为你的专用变量。在变量名前面缀以"$"，表示它是一个变量，而不是一个字面字符串。

```
<!--#set var="modified" value="$LAST_MODIFIED" -->
```

若要在字面字符串中使用"$"，必须使用转义符号"$"：

```
<!--#set var="cost" value="\$100" -->
```

最后，如果要在较长的字符串中使用变量，可以用花括号把变量名括起来，以免变量名与其他字符混淆(要对这种情况举例说明有点难度，但还是希望你能领会)。

```
<!--#set var="date" value="${DATE_LOCAL}_${DATE_GMT}" -->
```

条件表达式

有了变量，就可以设置和比较它们的值以表示条件，SSI也因此成为一种简洁的编程语言。 `mod_include` 提供了 `if` ， `elif` ， `else` ， `endif` 等结构以构造条件语句，从同一个页面高效地产生多个逻辑页面。

条件结构如下：

```
<!--#if expr="test_condition" -->
<!--#elif expr="test_condition" -->
<!--#else -->
<!--#endif -->
```

*test_condition*可以是任何逻辑比较：可以是一个值和另一个值比较，也可以是测试一个特定的值是否为"真"(一个给定的字符串如果非空则为真)。完整的比较操作符列表，参见 `mod_include` 。以下是可能会用到的一些例子。

在配置文件中，可以这样设置：

```
BrowserMatchNoCase macintosh Mac
BrowserMatchNoCase MSIE InternetExplorer
```

如果客户端在Macintosh上运行Internet Explorer，则上例设置环境变量"Mac"和"InternetExplorer"都为真。

然后，在允许SSI的文档中，可以这样设置：

```
<!--#if expr="${Mac} && ${InternetExplorer}" -->
Apologetic text goes here
<!--#else -->
Cool JavaScript code goes here
<!--#endif -->
```

我一点也不反对在Mac上运行IE，只是上个星期我花了好几个小时试图在Mac上的IE中使用JavaScript，而它在其他地方都能正常运作，以上只是一个临时的妥协方案。

任何其他变量(或者是你定义的，或者是标准的环境变量)都可以用于条件语句。利用Apache的 `SetEnvIf` 以及其他相关指令设置环境变量，此功能可以很好地实现动态页面而无须借助于CGI。

总结

SSI固然不能替代CGI或者其他动态页面技术，但它是在页面中插入众多小型的动态片段的优秀方法，而无须大量额外的操作。

.htaccess文件

`.htaccess` 文件提供了针对每个目录改变配置的方法。

.htaccess 文件

相关模块

- `core`
- `mod_authn_file`
- `mod_authz_groupfile`
- `mod_cgi`
- `mod_include`
- `mod_mime`

相关指令

- `AccessFileName`
- `AllowOverride`
- `Options`
- `AddHandler`
- `SetHandler`
- `AuthType`
- `AuthName`
- `AuthUserFile`
- `AuthGroupFile`
- `Require`

工作原理和使用方法

`.htaccess` 文件(或者"分布式配置文件")提供了针对每个目录改变配置的方法，即在一个特定的目录中放置一个包含指令的文件，其中的指令作用于此目录及其所有子目录。

说明：

如果需要使用 `.htaccess` 以外的其他文件名，可以用 `AccessFileName` 指令来改变。例如，需要使用 `.config` ，则可以在服务器配置文件中按以下方法配置：

```
AccessFileName .config
```

通常， `.htaccess` 文件使用的配置语法和主配置文件一样。 `AllowOverride` 指令按类别决定了 `.htaccess` 文件中哪些指令才是有效的。如果一个指令允许在 `.htaccess` 中使用，那么在本手册的说明中，此指令会有一个覆盖项段，其中说明了为使此指令生效而必须在 `AllowOverride` 指令中设置的值。

例如，本手册对 `AddDefaultCharset` 指令的阐述表明此指令可以用于 `.htaccess` 文件中(见"作用域"项)，而覆盖项一行是 `FileInfo` ，那么为了使 `.htaccess` 中的此指令有效，则至少要设置 `AllowOverride FileInfo` 。

例子：

作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo

如果不能确定某个指令是否可以用于 `.htaccess` 文件，可以查阅手册中对指令的说明，看在"作用域"行中是否有".htaccess" 。

(不)使用.htaccess文件的场合

一般情况下，不应该使用 `.htaccess` 文件，除非你对主配置文件没有访问权限。有一种很常见的误解，认为用户认证只能通过 `.htaccess` 文件实现，其实并不是这样，把用户认证写在主配置文件中是完全可行的，而且是一种很好的方法。

`.htaccess` 文件应该被用在内容提供者需要针对特定目录改变服务器的配置而又没有root权限的情况下。如果服务器管理员不愿意频繁修改配置，则可以允许用户通过 `.htaccess` 文件自己修改配置，尤其是ISP在同一个机器上运行了多个用户站点，而又希望用户可以自己改变配置的情况下。

虽然如此，一般都应该尽可能地避免使用 `.htaccess` 文件。任何希望放在 `.htaccess` 文件中的配置，都可以放在主配置文件的 `<Directory>` 段中，而且更高效。

避免使用 `.htaccess` 文件有两个主要原因。

首先是性能。如果 `AllowOverride` 启用了 `.htaccess` 文件，则Apache需要在每个目录中查找 `.htaccess` 文件，因此，无论是否真正用到，启用 `.htaccess` 都会导致性能的下降。另外，对每一个请求，都需要读取一次 `.htaccess` 文件。

还有，Apache必须在所有上级的目录中查找 `.htaccess` 文件，以使所有有效的指令都起作用(参见指令的生效)，所以，如果请求 `/www/htdocs/example` 中的页面，Apache必须查找以下文件：

```
/.htaccess

/www/.htaccess

/www/htdocs/.htaccess

/www/htdocs/example/.htaccess
```

总共要访问4个额外的文件，即使这些文件都不存在。(注意，这可能仅仅由于允许根目录"`/`"使用`.htaccess`，虽然这种情况并不多。)

其次是安全。这样会允许用户自己修改服务器的配置，这可能会导致某些意想不到的修改，所以请认真考虑是否应当给予用户这样的特权。但是，如果给予用户较少的特权而不能满足其需要，则会带来额外的技术支持请求，所以，必须明确地告诉用户已经给予他们的权限，说明`AllowOverride`设置的值，并引导他们参阅相应的说明，以免日后生出许多麻烦。

注意，在`/www/htdocs/example`目录下的`.htaccess`文件中放置指令，与在主配置文件中`<Directory /www/htdocs/example>`段中放置相同指令，是完全等效的。

`/www/htdocs/example`目录下的`.htaccess`文件：

`/www/htdocs/example`目录下的.htaccess文件的内容：

```
AddType text/example .exm
```

`httpd.conf`文件中摘录的内容：

```
<Directory /www/htdocs/example>

AddType text/example .exm
</Directory>
```

但是，把配置放在主配置文件中更加高效，因为只需要在Apache启动时读取一次，而不是在每次文件被请求时都读取。

将`AllowOverride`设置为`none`可以完全禁止使用`.htaccess`文件：

```
AllowOverride None
```

指令的生效

`.htaccess`文件中的配置指令作用于`.htaccess`文件所在的目录及其所有子目录，但是很重要的、需要注意的是，其上级目录也可能会有`.htaccess`文件，而指令是按查找顺序依次生效的，所以一个特定目录下的`.htaccess`文件中的指令可能会覆盖其上级目录中的`.htaccess`文

件中的指令，即子目录中的指令会覆盖父目录或者主配置文件中的指令。

例子：

`/www/htdocs/example1` 目录中的 `.htaccess` 文件有如下内容：

```
Options +ExecCGI
```

(注意：必须设置" `AllowOverride Options` "以允许在 `.htaccess` 中使用" `Options` "指令)

`/www/htdocs/example1/example2` 目录中的 `.htaccess` 文件有如下内容：

```
Options Includes
```

由于第二个 `.htaccess` 文件的存在， `/www/htdocs/example1/example2` 中的CGI执行是不允许的，而只允许 `Options Includes` ，它完全覆盖了之前的设置。

将.htaccess合并到主配置文件中

正如在配置段(容器)中讨论的那样， `.htaccess` 文件能够覆盖 `<Directory>` 段中对相应目录的设置，但是也同样会被主配置文件中其它类型的配置段所覆盖。这个特性可以用来强制实施某些配置，甚至在 `AllowOverride` 已经许可的情况下。举个例子来说，为了强迫在 `.htaccess` 中禁止脚本执行但不限制其它的情况下，可以这样：

```
<Directory />

 Allowoverride All
</Directory>

<Location />

 Options +IncludesNoExec -ExecCGI
</Location>
```

认证举例

如果你只是为了知道如何认证，而直接从这里开始看的，有很重要的一点需要注意，有一种常见的误解，认为实现密码认证必须要使用 `.htaccess` 文件，其实是不正确的。把认证指令放在主配置文件的 `<Directory>` 段中是一个更好的方法，而 `.htaccess` 文件应该仅仅用于无权访问主配置文件的时候。参见上述关于何时应该与何时不应该使用 `.htaccess` 文件的讨论。

有此声明在先，如果你仍然需要使用 `.htaccess` 文件，请继续看以下说明。

`.htaccess` 文件的内容：

```
AuthType Basic
AuthName "Password Required"
AuthUserFile /www/passwords/password.file
AuthGroupFile /www/passwords/group.file
Require Group admins
```

必须设置 `AllowOverride AuthConfig` 以允许这些指令生效。

更详细的说明，请参见认证、授权、访问控制。

服务器端包含(SSI)举例

`.htaccess` 文件的另一个常见用途是允许一个特定的目录使用服务器端包含(SSI)，可以在需要的目录中放置 `.htaccess` 文件，并作如下配置：

```
Options +Includes
AddType text/html shtml
AddHandler server-parsed shtml
```

注意，必须同时设置 `AllowOverride Options` 和 `AllowOverride FileInfo` 以使这些指令生效。

更详细的有关服务器端包含的说明，请参见SSI指南。

CGI举例

可以通过 `.htaccess` 文件允许在特定的目录中执行CGI程序，需要作如下配置：

```
Options +ExecCGI
AddHandler cgi-script cgi pl
```

另外，如下配置可以使给定目录下的所有文件被视为CGI程序：

```
Options +ExecCGI
SetHandler cgi-script
```

注意，必须同时设置 `AllowOverride Options` 和 `AllowOverride FileInfo` 以使这些指令生效。

更详细的有关CGI编程和配置的说明，请参见CGI指南。

疑难解答

如果在 `.htaccess` 文件中的某些指令不起作用，可能有多种原因。

最常见的原因是 `AllowOverride` 指令没有被正确设置，必须确保没有对此文件区域设置 `AllowOverride None` 。有一个很好的测试方法，就是在 `.htaccess` 文件随便增加点无意义的垃圾内容，如果服务器没有返回了一个错误消息，那么几乎可以断定设置了 `AllowOverride None` 。

在访问文档时，如果收到服务器的出错消息，应该检查Apache的错误日志，可以知道 `.htaccess` 文件中哪些指令是不允许使用的，也可能会发现需要纠正的语法错误。

用户网站目录

在多用户系统中，用 `UserDir` 指令可以允许每个用户在其宿主目录中拥有一个网络站点。使用URL `http://example.com/~username/` 的访问者可以获得用户" `username` "的宿主目录中的内容或者用 `UserDir` 指定的子目录中的内容。

用户网站目录

相关模块

- `mod_userdir`

相关指令

- `UserDir`
- `DirectoryMatch`
- `AllowOverride`

用UserDir设置文件路径

可以用 `UserDir` 指令指定被网络读取的用户网站目录。此指令有几种不同的形式。

如果路径没有前导斜杠，则被当作该用户宿主目录下的子目录。如果有以下配置：

```
UserDir public_html
```

则URL `http://example.com/~rbowen/file.html` 会被解释为文件路径：`/home/rbowen/public_html/file.html`

如果路径有前导斜杠，则用此路径和用户名构造路径。如果有以下配置：

```
UserDir /var/html
```

则URL `http://example.com/~rbowen/file.html` 会被解释为文件路径：`/var/html/rbowen/file.html`

如果路径中有星号(*)，则星号部分会被用户名所取代。如果有以下配置：

```
UserDir /var/www/*/docs
```

则URL `http://example.com/~rbowen/file.html` 会被解释为文件路径：`/var/www/rbowen/docs/file.html`

限定哪些用户可以使用此功能

使用 `UserDir` 可以限定被允许使用此功能的用户：

```
UserDir enabled

UserDir disabled root jro fish
```

上述配置使除了列在 `disabled` 中的用户以外的所有用户都可以使用此功能。还可以禁止所有用户而只允许部分用户使用此功能，例如：

```
UserDir disabled

UserDir enabled rbowen krietz
```

更多的例子请参见 `UserDir` 文档。

启用对每个用户都有效的cgi目录

`<Directory>` 指令可以指定每个用户主目录中的一个特定的目录为"允许cgi"的目录，使每个用户都可以拥有自己的 `cgi-bin` 目录。

```
<Directory /home/*/public_html/cgi-bin/>

 Options ExecCGI

 SetHandler cgi-script

 </Directory>
```

这样"放肆的"设置使得 `UserDir` 被设置成 `public_html` ，CGI程序 `example.cgi` 可以用下面的URL从那个目录加载：

```
http://example.com/~rbowen/cgi-bin/example.cgi
```

允许用户改变配置

用户可以通过 `.htaccess` 文件改变其网络空间的服务器配置，因此必须确保 `AllowOverride` 指令被正确设置，以限定用户只能使用被允许的指令。其细节请参见.htaccess指南。

针对特定平台的说明

Microsoft Windows

Apache的使用

此文阐述如何在Windows平台上安装、配置和运行Apache2.0

参见：在Microsoft Windows上使用Apache

编译Apache

此文会指出在Windows平台上编译Apache以前必须了解的许多要点。

参见：在Microsoft Windows平台上编译Apache

其他平台

Novell NetWare

此文阐述如何在Novell NetWare 5.1及其更新版本中安装、配置和运行Apache2.0

参见：在Novell NetWare平台上使用Apache

HP-UX

此文阐述如何在HP-UX中运行Apache

参见：在HP-UX中运行Apache

EBCDIC

Apache HTTP服务器的1.3版本是第一个支持使用EBCDIC作为宿主编码集的非ASCII体系主机的版本。

警告： 这个文档中，不包含Apache HTTP服务器2.0更新的内容，有些内容可能仍然有效，使用中请注意。

参见：The Apache EBCDIC Port

在Microsoft Windows中使用Apache

此文档阐述了如何在Microsoft Windows平台上安装、配置和运行Apache 2.0 。 如果你发现了bug，或者希望以其他方式作出贡献，请使用我们的bug报告页面。

本文档的大多数内容假定你是从一个二进制发布版安装Apache到Windows上。 如果你想自己编译Apache(可能有助于开发和跟踪bugs)，参看编译Windows下的Apache。

基于当前的**Windows**版本状况，本文档使用到的缩写及其意义解释包括：

- **Windows NT:** 指所有基于NT核心的Microsoft Windows操作系统的版本，包括Windows NT, Windows 2000, Windows XP, Windows.NET Server 2003及后续版本。
- **Windows 9x:** 指所有定位于家庭使用的Microsoft Windows操作系统的版本，包括Windows 95 , Windows 98, Windows ME 。

对操作系统的要求

Apache 2.0被设计为在Windows NT上运行。它的二进制安装程序只能在x86处理器上运行，比如Intel和AMD的芯片。Apache可能也能够运行在Windows 9x上，但是并没有经过测试，也不建议在实际工作的系统上这样使用。

任何情况下都必须确保TCP/IP网络协议已经安装。如果在Windows 95上运行，必须安装"Winsock2"升级补丁。 "Winsock2" for Windows 95可以在这里下载。

如果在NT 4.0上运行，建议安装Service Pack 6 ，因为Service Pack 4有众所周知的TCP/IP和Winsock完整性的问题，在以后的Service Pack中解决了这些问题。

下载 Apache for Windows

关于Apache最新版本的有关信息可以在http://httpd.apache.org/download.cgi上找到。那里会列出当前发行版本、 所有最近的alpha和beta测试版本以及镜像web站点和匿名ftp服务器的信息。

你应该下载带有 `.msi` 扩展名的Apache for Windows版本。 这是一个单一的Microsoft Installer文件，包含了Apache ， 可以立即安装并运行。 还有一个单独的 `.zip` 文件只打包了源码， 可以用Microsoft Visual C++ (Visual Studio)工具来编译。

安装 Apache for Windows

Apache的安装需要有Microsoft Installer 1.2或更高版本。 在Windows 9x中，你可以从这里升级Microsoft Installer到2.0， 在Windows NT 4.0和2000中，2.0的升级版在这里。Windows XP/2003不需要这个升级。

注意，使用这个安装包不能在同一个机器上安装两套Apache 2.0。 但是，在同一个机器上，安装一个1.3系列和一个2.0系列的Apache则没有问题。 如果你需要在同一个机器上安装两套不同的2.0版本， 则必须用代码编译和安装Apache

运行已下载的上述Apache `.msi` 文件。安装程序会要求提供下列信息：

1. **Network Domain** 你的服务器已经或者将要注册的DNS域名。比如你的服务器的全称DNS域名是 `server.mydomain.net` ，你应当在这里输入： `mydomain.net`
2. **Server Name** 你的服务器的全称DNS域名，如上情况你应当在这里输入： `server.mydomain.net`
3. **Administrator's Email Address** 服务器管理员的email地址。这个地址将会在默认的出错页面上显示给客户端。
4. **For whom to install Apache** 如果你希望Apache在80端口监听，并被安装为服务(即使无人登陆，Apache仍将运行)，就选择" `for All Users, on Port 80, as a Service - Recommended` "；如果你希望将Apache安装为个人试验使用，或者已经有一个运行于80端口的WWW服务器，就选择" `only for the Current User, on Port 8080, when started Manually` "。
5. **The installation type** 选择 `Typical` 会安装除开发模块需要使用的源码和库以外的所有内容。选择 `Custom` 可以自定义安装哪些项目。完整安装大约需要13MB磁盘空间，这其中并不包含你的网站文件所用空间。
6. **Where to install** 安装Apache的文件夹，默认为 `C:\Program Files\Apache Group` 文件夹下的 `Apache2` 子文件夹。

安装期间，Apache将会配置你所选择的安装目录下的 `conf` 文件夹中的文件。 但是如果那个目录下已有同名文件存在，原有文件将不会被覆盖， 而相应的新文件将会被加上 `.default` 扩展名。 所以，举例来说，如果 `conf\httpd.conf` 已经存在，那么不会对它做任何改变， 而新版本 `conf\httpd.conf` 的内容将会被写入文件 `conf\httpd.conf.default` 。 安装完成以后你应该检查 `.default` 文件中的内容看看有没有不同， 如果必要，更新你原有的配置文件。

而且，如果你已经有一个名为 `htdocs\index.html` 的文件，它不会被覆盖掉 (也不会安装 `index.html.default` 文件)。 这意味着你在一个旧版本Apache上安装新版本是安全的 (但是你必须在安装之前首先停掉原有服务器然后在安装完成后重新启动它)。

安装Apache以后，你应该编辑 `conf` 目录下的配置文件。这些文件已在安装期间被配置好以便Apache能够从安装目录运行，文档目录被配置为安装目录下的子目录 `htdocs` 。在你开始真正使用之前还有很多选项需要设置。但是为了尽快开始，可以使用安装时自动配置的配置

文件。

配置 Apache for Windows

与Unix系统下的版本一样，Apache使用 `conf` 目录下的文件进行配置，但是Windows版本有几个不同的指令，参见指令索引察看全部可用指令。

Apache for Windows主要的不同点是：

- 因为Apache for Windows是多线程的，它并不像Unix版本那样为每个请求使用一个单独的进程。 而是通常运行两个Apache进程：一个父进程，和一个处理请求的子进程。在子进程内部由多个单独的线程来处理每个请求。

  因此与进程管理相关的指令是不同的：

  `MaxRequestsPerChild` 就像Unix版本中的指令一样，这条指令控制一个进程退出前将为多少个请求提供服务。 然而，与Unix不同的是，一个进程将为所有请求而不是只为一个请求服务，因此如果设置这条指令，建议将它设为一个很大的值。默认设置 `MaxRequestsPerChild 0` 使得进程从不退出。

  警告： 启动新的子进程时将会重新读入服务器配置文件。如果你修改了 **`httpd.conf`** ，新的子进程将有可能不能启动或者可能得到预期之外的结果。

  `ThreadsPerChild` 是一条新的指令，用来告诉服务器应该使用多少个线程，指明了服务器可以立刻处理的最大连接数；如果你的站点有大量的点击，请确认你设置了足够大的值。推荐的默认设置是 `ThreadsPerChild 50` 。

- 接收文件名作为参数的指令必须使用Windows文件名而不是Unix文件名。 但是，因为Apache内部使用Unix风格的名字，你必须使用正斜杠而不是反斜杠。可以使用盘符；如果省略盘符，将假定使用Apache可执行文件所在盘符。

- Apache for Windows具有运行时装入模块的能力，不需要重新编译。如果Apach在正常情况下编译，它会在 `\Apache2\modules` 目录下安装许多可选模块。 要激活它们或其他模块，必须使用新的 `LoadModule` 指令。 举例来说，要激活状态模块，使用下列指令(除了 `access.conf` 中的状态激活指令以外)：

  ```
  LoadModule status_module modules/mod_status.so
  ```

  也可使用创建可加载模块中的信息。

- Apache也可以加载ISAPI(Internet Server Applications Programming Interface)扩展，例如被Microsoft IIS服务器和其他一些Windows服务器所使用的。这里有更多相关信息。注意Apache不能加载ISAPI 过滤器。

- 当运行CGI脚本时，Apache查找脚本解释器是由 `ScriptInterpreterSource` 指令配置的。

- 由于在Windows下管理具有像 `.htaccess` 这样名字的文件是很困难的，你会发现在配置文件中使用 `AccessFilename` 指令改变它的文件名是很有用的。
- 在Windows NT上，Apache启动时发生的错误将会记入Windows事件日志(event log)。这个机制将在Apache尚不能使用 `error.log` 文件的时候运作。你可以通过"事件查看器"的MMC接口查看Windows事件日志。

  注意，在**Windows 9x**上不存在事件日志机制，因此无法记录启动错误。

以服务方式运行Apache for Windows

Apache仅能够在Windows NT上作为服务运行。

你可以选择在安装Apache时自动将其安装为一个服务。如果你选择"for all users"，那么Apache将会被安装为服务。如果你选择了"only for the Current User"，你可以在安装后手动将Apache注册为服务。注意，你必须是Administrators组的成员才能成功注册服务。

使用Apache Service Monitor工具，可以查看和管理你所在网络上的所有机器上安装的Apache服务的状态。为了能够使用这个工具管理Apache服务，你必须首先自动或手动安装Apache服务。

你可以在Apache安装目录的 `bin` 子目录下，使用如下命令将Apache安装为Windows NT服务：

```
apache -k install
```

如果你想指定服务的名称，可以使用下面的命令。当你在同一机器上安装多个Apache服务时，你必须为它们指定不同的名字。

```
apache -k install -n "服务名"
```

如果你想为不同名称的服务使用不同的配置文件，则安装时需要指定配置文件：

```
apache -k install -n "服务名" -f "c:\files\my.conf"
```

如果你使用的是第一个命令，也就是除 `-k install` 外没有其它命令行参数，那么被安装的服务名称将是： `Apache2` ，配置文件将使用 `conf\httpd.conf` 。

要移除一个Apache服务很简单：

```
apache -k uninstall
```

或者使用下述命令移除特定名称的Apache服务：

```
apache -k uninstall -n "服务名"
```

通常，启动、重启、关闭Apache服务的方法是使用Apache Service Monitor工具，另外也可以使用控制台命令：`NET START Apache2` 和 `NET STOP Apache2` 或者通过Windows服务控制面板。在启动Apache服务之前，你应当使用下面的命令检查一下配置文件的正确性：

```
apache -n "服务名" -t
```

你可以通过命令行开关来控制Apache服务。要启动一个已经安装的Apache服务，可以使用：

```
apache -k start
```

要停止一个已经安装的Apache服务，可以使用：

```
apache -k stop
```

或

```
apache -k shutdown
```

要重启一个运行中的Apache服务，强制它重新读取配置文件，可以使用：

```
apache -k restart
```

默认情况下，Apache服务将被注册为以本地系统用户(`LocalSystem` 帐号)身份运行。 `LocalSystem` 帐号没有网络权限，不能通过任何Windows安全机制访问网络，包括文件系统、命名管道、DCOM或secure RPC，但是它对于本地资源却拥有广泛的特权。

永远不要把网络权限授予 **`LocalSystem`** 帐号！如果你需要**Apache**能够访问网络资源，最好按照下述方法为**Apache**另外建立一个单独的帐号。

你应该建立一个单独的帐号来运行Apache服务。特别是在必须通过Apache访问网络资源的时候，我们更加强烈建议你这样做。

1. 创建一个普通域用户帐号，并牢记对应的密码。
2. 授予这个新建的帐号 `作为服务登陆` 和 `作为操作系统一部分运行` 权限。在Windows 2000/XP/2003上你可以使用"组策略"或通过"本地安全策略"的MMC接口来完成这个操作。
3. 确认新建的帐号是Users组的一个成员。
4. 确认新建的帐号具有读取和执行(RX)所有文档和脚本目录(例如：`htdocs` 和 `cgi-bin`)的权限。
5. 确认新建的帐号对Apache的 `logs` 目录具有读/写/删除(RWD)的权限。

6. 确认新建的帐号对 `Apache.exe` 二进制文件具有读取和执行(RX)的权限。

一个很好的实践经验是赋予运行Apache服务的用户读取和执行(RX)整个Apache2目录的权限，并且对 `logs` 子目录具有读/写/删除(RWD)的权限。

如果你允许使用这个帐号作为一个用户和服务登录，你就可以用这个帐号登录上去测试执行脚本、读取web页的权限，还可以通过控制台窗口启动Apache 。如果这样工作正常，你又执行了上述的操作，那么Apache就能够正常地作为服务运行了。

错误代码**2186**是一个很好的提示，说明你需要检查"登陆为"选项，因为服务器不能访问必要的网络资源。

当启动Apache服务时你可能会遇到一个来自Windows服务管理器的错误信息。 例如，如果你想使用控制面板中的服务小程序启动Apache，可能会得到下面这条信息：

```
Could not start the Apache2 service on \\COMPUTER

Error 1067; The process terminated unexpectedly.
```

只要启动Apache出错你就会得到这个错误信息。为了弄清是什么引起了错误， 你应该遵循作为控制台程序运行Apache中的建议。

对于让Apache在Windows 9x下以类似Windows NT服务的方式运行有一些支持。都是高度试验性的，即使能够工作，Apache软件基金会将不会证实其可靠性和未来的支持。继续进行你自己的冒险吧！

这两种"服务"有相当大的区别：

如果你输入下列命令，Apache会尝试启动，如果成功他将在后台运行：

```
Apache -n "服务名" -k start
```

例如，通过桌面的快捷方式运行，如果服务启动成功一个控制台窗口会快速闪过并立刻消失。如果启动时Apache检测到比如 `httpd.conf` 文件中有不正确的设置这样的问题，则控制台窗口会保持可见。这样将显示一个有助于追踪错误原因的错误信息。

Windows 9x不支持 `NET START` 或者 `NET STOP` 命令，因此你在命令中必须使用Apache的服务控制选项。你可能希望为每个命令设置一个快捷方式以便你能够只需要从开始菜单或者桌面一点就能执行所需的操作。

Apache和Windows 9x没有提供让Apache服务以特定的具有网络权限的用户身份运行的支持。实际上，Windows 9x在本地机器上根本没有提供安全性。这就是Apache软件基金会从不支持使用Windows 9x作为公用httpd服务器的原因。这个便利存在的唯一目的是协助用户开发web内容和学习Apache服务器，或者也许在一个安全的、私有的网络上充当intranet服务器。

作为控制台程序运行Apache

虽然通常推荐将Apache作为服务来运行，但是在某些情况下从命令行运行反而更加容易。在Windows 9x上，从命令行运行Apache是推荐的方法，因为这些系统中缺乏可靠的服务支持。

可以使用下列命令将Apache作为控制台程序从命令行运行：

```
apache
```

Apache将会一直保持运行，直到被Ctl+C组合键中断。

你还可以在安装后通过 `开始按钮 --> 程序 --> Apache HTTP Server 2.2.xx --> Control Apache Server` 在控制台中运行Apache 。这将会打开一个控制台窗口并在其中启动Apache 。如果你没有将Apache安装为服务，该窗口将一直保持打开，直到被Ctl+C组合键中断，并在几秒钟后退出。如果你已经将Apache安装为服务，那么那个快捷方式将会启动服务，如果Apache服务已经启动了，则什么也不做。

你可以在另外一个控制台窗口中输入以下命令停止正在运行的Apache服务：

```
apache -k shutdown
```

这种方法比使用Ctl+C组合键更好，因为它可以让Apache完成所有当前的操作并且优雅的清理所占用的资源。

Apache可以被从新启动，这将导致它重新读取配置文件，重启前所有正在进行的操作都将不被打断的完成。可以使用下面的命令重启Apache ：

```
apache -k restart
```

请熟悉Unix版本Apache的用户注意，这些命令对应于Unix命令 `kill -TERM _pid_` 和 `kill -USR1 _pid_` 。命令中之所以使用 `-k` 选项，就是为了提醒用户这是源自Unix下的 `kill` 命令。

如果Apache控制台窗口在启动后出乎意料的立即关闭，请打开一个新的控制台窗口，切换到Apache的安装目录下bin子目录中，运行 `apache` 命令并读取错误信息。然后再到日志目录中查看 `error.log` 文件以寻找可能的配置错误。如果你的Apache采用的是默认安装，这些命令应当是：

```
c:

cd "\Program Files\Apache Group\Apache2\bin"

apache
```

等候Apache停止，或者使用Ctl+C组合键，然后输入下列内容：

```
cd ..\logs

more < error.log
```

可以在命令行中为Apache指定一个配置文件，有两种方法可以在命令行中指定配置文件：

- `-f` 可以指定一个绝对路径或相对路径的配置文件：

  ```
  apache -f "c:\my server files\anotherconfig.conf"
  ```

  或

  ```
  apache -f files\anotherconfig.conf
  ```

- `-n` 可以指定已安装的Apache服务所使用的配置文件：

  ```
  apache -n "服务名"
  ```

在这两种情况下，必须在配置文件中设置正确的 `ServerRoot` 值。

如果你没有使用 `-f` 或 `-n` 指定配置文件的路径，Apache将会使用硬编码在服务器内的路径，比如：`conf\httpd.conf` 。这个内置的路径是相对于安装目录的，要想检查这个内置的路径，你可以通过 `-V` 开关调用Apache， 查看名为 `SERVER_CONFIG_FILE` 的变量值：

```
apache -V
```

Apache将会按照下列顺序检查 `ServerRoot` 值：

1. 通过 `-C` 命令行开关指定的 `ServerRoot` 值。
2. 通过 `-d` 命令行开关指定的值。
3. 当前工作目录。
4. 安装二进制文件时在注册表中登记的项目。
5. 编译进二进制文件的值，该值默认为：`/apache` ，你可以使用 `apache -V` 命令查看显示出来的 `HTTPD_ROOT` 变量的值。

在安装过程中，将会在注册表中新建一个版本特定的注册表键。这个键的位置取决于安装类型。如果你选择的是"for all users"，那么将位于 `HKEY_LOCAL_MACHINE` 分支下，如下所示：

```
HKEY_LOCAL_MACHINE\SOFTWARE\Apache Group\Apache\2.0.43
```

如果你选择的是"for the current user only"，那么将位于 `HKEY_CURRENT_USER` 分支下，其中的内容取决于当前登陆的用户，如下所示：

```
HKEY_CURRENT_USER\SOFTWARE\Apache Group\Apache\2.0.43
```

这个键已经被编译进了服务器，使得你可以测试新版本而又不影响老版本。当然，你必须注意不要在同一个目录中安装两个不同的版本。

如果你没有使用二进制安装，在某些情况下Apache将会抱怨注册表键丢失。如果在这种情况下服务器仍然可以找到正确的配置文件，那么就可以安全的忽略它。

这个注册表键其实就是包含 `conf` 子目录的 `ServerRoot` 目录。Apache将要从中读取 `httpd.conf` 文件。如果该配置文件中又包含了一个 `ServerRoot` 指令，并且指向的目录与注册表中登记的不同，Apache将以配置文件中的指令为准。如果你复制了配置文件或者整个Apache目录到一个新的位置，你要千万记得修改 `httpd.conf` 中的 `ServerRoot` 指令，使其指向正确的位置。

安装的测试

启动Apache运行以后(不管是控制台窗口还是作为服务)， 它会在80端口上进行监听(除非你改变了配置文件中的 `Listen` 指令。 要连接到服务器访问默认页面，启动一个浏览器并输入下列URL ：

```
http://localhost/
```

应该出现一个欢迎页面，并且页面上有到Apache用户手册的链接。 如果什么都没有发生或是得到了一个错误，检查 `logs` 子文件夹中的 `error.log` 文件。如果你的主机没有联网或者DNS配置有严重问题，你也许需要输入这样的URL ：

```
http://127.0.0.1/
```

如果你将Apache配置为在非80端口监听(比如：8080)，你应当使用下面的URL明确指定端口：

```
 http://127.0.0.1:8080/
```

一旦你的基本配置可以工作了，你应该编辑 `conf` 目录下的文件来恰当地配置Apache 。 此外，如果你改变了作为NT服务运行的Apache的配置，你应该首先尝试从命令行启动来保证能够正确地启动Apache服务。

因为Apache不能与其他TCP/IP应用程序共享同一端口，你可能需要先停止或者卸载或者重新配置某些特定的服务。这包括(但不限于)别的web服务器和BlackIce那样的防火墙。如果你只能在禁止其他服务的情况下启动Apache，那么需要重新配置Apache或者其他程序使它们不监听同一个TCP/IP端口。

在Microsoft Windows上编译Apache

在你开始编译Apache之前有许多重要问题需要注意。开始之前请先看看在Microsoft Windows上使用Apache 。

系统要求

编译Apache需要正确安装以下环境：

- 磁盘空间

  确保至少有50MB空闲磁盘空间可用。安装以后Apache使用大约10MB磁盘空间，再加上会快速增长的日志和缓存文件需要的空间。 实际需要的空间大小会在相当大程度上取决于你选择的配置以及使用的第三方模块和库。

- Microsoft Visual C++ 5.0 或更高版本

  可以使用命令行工具，也可以在Visual Studio集成开发环境内编译Apache。使用命令行工具要求环境变量中包含 `PATH` , `INCLUDE` , `LIB` 和其他一些变量，这些环境变量可以用 `vcvars32` 批处理文件来设置：

  ```
  "c:\Program Files\DevStudio\VC\Bin\vcvars32.bat"
  ```

- Windows Platform SDK

  Visual C++ 5.0 编译需要一套新版的Microsoft Windows Platform SDK来允许Apache的某些特性。 对于命令行编译，用 `setenv` 批处理文件来设置环境变量：

  ```
  "c:\Program Files\Platform SDK\setenv.bat"
  ```

  随Visual C++ 6.0 及以后版本发布的Platform SDK文件足以满足要求，所以新版本的用户可以略过这个要求。

  注意，需要新版的Windows Platform SDK来使得Apache支持的全部 `mod_isapi` 特性可用。没有新版SDK的话，在 MSVC++ 5.0 下编译Apache会出现某些 `mod_isapi` 特性将被禁止的警告。在http://msdn.microsoft.com/downloads/sdks/platform/platform.asp可以找到新版的Microsoft Winodws Platform SDK。

- awk工具(awk, gawk或类似软件)

为了在编译系统内安装Apache，用 `awk.exe` 工具修改了几个文件。选择awk是因为它很小，易于下载(与Perl或者WSH/VB相比)，而且能够完成生成文件的任务。Brian Kernighan的http://cm.bell-labs.com/cm/cs/who/bwk/站点有一个编译好的本地Win32代码版本，这个文件http://cm.bell-labs.com/cm/cs/who/bwk/awk95.exe你必须将它名字保存为 `awk.exe` 而不是 `awk95.exe` 。

注意Developer Studio集成开发环境只能在Tools - Options菜单中的Directories页上列出的可执行文件搜索路径列表中查找 `awk.exe` (对于Developer Studio 7.0 是在the Projects - VC++ Directories 面板)。 把 `awk.exe` 的路径加入到列表中，并按要求加入到系统 `PATH` 环境变量里。如果你用的是Cygwin (http://www.cygwin.com/)需要注意，awk工具的文件名是 `gawk.exe` 而文件 `awk.exe` 实际上是 `gawk.exe` 的一个符号连接。 而Windows命令行解释程序不认识符号连接，因此编译二进制安装文件会失败。可行的变通办法是从cygwin安装目录删除文件 `awk.exe` 并把 `gawk.exe` 改名为 `awk.exe` 。

- [可选] OpenSSL库(因为 `mod_ssl` 和 `ab.exe` 用到ssl支持)

  警告：在整个世界范围使用和发布高强度密码体系与专利知识产权都有相当大的限制和严格的禁令。OpenSSL包括了在美国及其他国家和地区受到出口条例、国内法律以及受专利保护的知识产权所限制的高强度密码体系。对于OpenSSL项目提供的代码，不管是Apache软件基金会还是OpenSSL项目都不能提供关于拥有、使用和发布该代码的法律建议。向你自己的法律顾问咨询，你需要为你自己的行为负责。

  为了编译 `mod_ssl` 或abs项目(`ab.exe` 用到SSL支持)，OpenSSL必须安装到 `srclib` 目录下名为 `openssl` 的子目录中，openSSL可以从http://www.openssl.org/source/获得。要是准备既编译 `release` 版本又编译 `debug` 版本，而且要禁止 0.9.7 版中受专利保护的特性，你应该使用下列编译命令：

```
perl Configure VC-WIN32

perl util\mkfiles.pl &gt;MINFO

perl util\mk1mf.pl dll no-asm no-mdc2 no-rc5 no-idea VC-WIN32 &gt;makefile

perl util\mk1mf.pl dll debug no-asm no-mdc2 no-rc5 no-idea VC-WIN32 &gt;mak

perl util\mkdef.pl 32 libeay no-asm no-mdc2 no-rc5 no-idea &gt;ms\libeay32.

perl util\mkdef.pl 32 ssleay no-asm no-mdc2 no-rc5 no-idea &gt;ms\ssleay32.

nmake

nmake -f makefile.dbg
```

- [可选] zlib源码 (用于 `mod_deflate`)

  Zlib必须安装到 `srclib` 目录下的 `zlib` 子目录，但是你不需要去编译那些源码。编译系统会直接把压缩源码编译到 `mod_deflate` 模块中去。 Zlib可以从http://www.gzip.org/zlib/获得 -- `mod_deflate` 已经经过验证可以使用版本 1.1.4 正确编译。

命令行编译

首先，将Apache源码包解压到合适的目录。打开一个命令提示符窗口并用 `cd` 切换到那个目录。

主要的Apache make文件命令都包含在文件 `Makefile.win` 中。要在Windows NT上编译Apache，只需要简单地使用下列命令之一就可以编译 `release` 或 `debug` 版本，分别是：

```
nmake /f Makefile.win _apacher

nmake /f Makefile.win _apached
```

两条命令都可以编译Apache 。后者会在编译结果文件中包含调试信息，使发现bugs和跟踪问题更容易。

Developer Studio集成开发环境的工作区编译

Apache也能够用VC++的Visual Studio集成开发环境编译。为了简化过程，提供了一个Visual Studio工作区文件： `Apache.dsw` 。 它阐述了完整的Apache二进制发行版需要的全部 `.dsp` 项目列表。 它包含了项目之间的依存关系来保证编译按合适的顺序进行。

打开 `Apache.dsw` 工作区文件，选择 `InstallBin` (根据需要选择编译 `Release` 或者 `Debug` 版本) 为活动项目。 `InstallBin` 会引发编译相关的项目并调用 `Makefile.win` 移动编译后的可执行文件和动态链接库。你可以改变 `InstallBin` 项目的设置来定制 `INSTDIR=` 选项，修改设置中General页里面的Build Command line条目。 `INSTDIR` 的缺省值是 `/Apache2` 目录。如果你只是想要测试编译(不安装)，就用 `BuildBin` 项目代替。

`.dsp` 项目文件使用Visual C++ 6.0格式发行。Visual C++ 5.0 (97)也能识别这种格式。而Visual C++ 7.0 (.net)必须把 `Apache.dsw` 和 `.dsp` 文件转换成 `Apache.sln` 和 `.msproj` 文件，如果有任何一个 `.dsp` 源文件改变了，必须重新转换相应的 `.msproj` 文件！ 这很容易，只需要在VC++ 7.0 集成开发环境中重新打开 `Apache.dsw` 文件。

Visual C++ 7.0 (.net)的用户还应该使用Build 菜单下的Configuration Manager对话框来不选中模块abs， `mod_ssl` 和 `mod_deflate` ， 对编译 `Debug` 和 `Release` 版本都是。 仅当 `srclib` 目录下至少存在 `openssl` 或者 `zlib` 子目录二者之一， 才能调用 `nmake` 或者明白地使用 `BinBuild` 目标直接从集成开发环境来编译这几个模块。

导出的那些 `.mak` 文件造成很大的争议，但对于 Visual C++ 5.0 的用户它们是编译 `mod_ssl` 、abs(带SSL支持的ab)和 `mod_deflate` 是必需的。 VC++ 7.0 (.net)的用户也能从中受益，用 `nmake` 编译比用 `binenv` 要快。 从VC++ 5.0 或 6.0 集成开发环境编译所有项目，再使用Project菜单 - Export导出所有make文件。 为了创建全部自动产生的动态目标你必须首先编译项目，以便互相之间的依存关系可以被正确解析。运行下面命令修正路径使之能编译到任何位置：

```
perl srclib\apr\build\fixwin32mak.pl
```

你必须在 `httpd` 源码树的顶层目录输入这个命令。 当前目录及其子目录下所有的 `.mak` 和 `.dep` 项目文件都将被改正，并且时间戳被调节到与 `.dsp` 一致。

如果你贡献修正项目文件的补丁，我们必须以Visual Studio 6.0 格式来确认项目文件。 改动应该简单而且只带有最少的编译和连接标记以便能够被从VC++ 5.0 到 7.0 的所有环境识别。

项目组件

`Apache.dsw` 工作区文件和 `makefile.win` `nmake` 脚本都是以下列顺序编译Apache服务器的 `.dsp` 项目文件：

1. `srclib\apr\apr.dsp`
2. `srclib\apr\libapr.dsp`
3. `srclib\apr-util\uri\gen_uri_delims.dsp`
4. `srclib\apr-util\xml\expat\lib\xml.dsp`
5. `srclib\apr-util\aprutil.dsp`
6. `srclib\apr-util\libaprutil.dsp`
7. `srclib\pcre\dftables.dsp`
8. `srclib\pcre\pcre.dsp`
9. `srclib\pcre\pcreposix.dsp`
10. `server\gen_test_char.dsp`
11. `libhttpd.dsp`
12. `Apache.dsp`

此外， `modules\` 子目录树包含了大多数模块的项目文件。

`support\` 子目录包含了一些附加程序的项目文件，它们运行时不是Apache的一部分， 但是管理员要使用它们来测试Apache和维护密码与日志文件。 Windows平台特有的支持项目在 `support\win32\` 目录下。

1. `support\ab.dsp`
2. `support\htdigest.dsp`
3. `support\htpasswd.dsp`
4. `support\logresolve.dsp`
5. `support\rotatelogs.dsp`
6. `support\win32\ApacheMonitor.dsp`
7. `support\win32\wintty.dsp`

一旦编译了Apache，它需要被安装在服务器根目录，缺省是在同一个盘符下的 `\Apache2` 目录。

要自动编译和安装所有文件到指定的目录*dir*，使用下列 `nmake` 命令之一：

```
nmake /f Makefile.win installr INSTDIR=_dir_

nmake /f Makefile.win installd INSTDIR=_dir_
```

`INSTDIR` 的*dir*参数给出了安装目录；如果要安装到 `\Apache2` 目录可以省略。

安装结果如下列：

- `_dir_\bin\Apache.exe` - Apache可执行文件
- `_dir_\bin\ApacheMonitor.exe` - 服务监视器托盘图表工具
- `_dir_\bin\htdigest.exe` - 摘要授权密码文件工具(Digest auth password file utility)
- `_dir_\bin\htdbm.exe` - SDBM授权数据库密码文件工具(SDBM auth database password file utility)
- `_dir_\bin\htpasswd.exe` - 基本授权密码文件工具(Basic auth password file utility)
- `_dir_\bin\logresolve.exe` - 日志文件dns名称查找工具
- `_dir_\bin\rotatelogs.exe` - 日志文件遍历工具
- `_dir_\bin\wintty.exe` - 控制台窗口工具
- `_dir_\bin\libapr.dll` - Apache可移植运行时共享库
- `_dir_\bin\libaprutil.dll` - Apache运行时共享库工具
- `_dir_\bin\libhttpd.dll` - Apache核心库
- `_dir_\modules\mod_*.so` - Apache可装载模块
- `_dir_\conf` - 配置目录
- `_dir_\logs` - 空日志目录
- `_dir_\include` - C语言头文件
- `_dir_\lib` - 连接库文件

关于从开发树编译Apache的警告

在每次发布 `发行` 版本之间，只有 `.dsp` 文件被维护。 考虑到会对审阅者的时间造成巨大浪费，并不重新产生 `.mak` 文件。 因此，你不能依靠上述的 `NMAKE` 命令来编译修订过的 `.dsp` 项目文件，除非你自己从项目中导出全部 `.mak` 文件。如果你在Microsoft Developer Studio环境中编译这样做是不必要的。同时注意在导出make文件之前编译 `BuildBin` 目标项目是非常值得的(或者用命令行目标 `_apacher` 或 `_apached`)。 许多文件在编译过程中自动产生。只有一次完全编译才提供为正确的编译行为编译正确的依存关系树所需要的全部依赖文件。

为创建供发布的 `.mak` 文件，一定要检查 `.mak` (或 `.dep`)中Platform SDK和其他头文件的依存性。 `DevStudio\SharedIDE\bin\` (VC5)或者 `DevStudio\Common\MSDev98\bin\` (VC6) 目录包含了 `sysincl.dat` 文件，其中列出了所有的例外情况来告诉VC++创建依存关系时不扫描列表中的文件， 更新此文件以包含这些头文件 (同时包括正斜杠和反斜杠路径，比

如 `sys/time.h` 和 `sys\time.h` 要同时列出)。 在发布的 `.mak` 文件中包含一个本地安装路径将使编译完全失败，所以不要忘了运行 `srclib/apr/build/fixwin32mak.pl` 来修正 `.mak` 文件中的绝对路径。

Using Apache With Novell NetWare

This document explains how to install, configure and run Apache 2.0 under Novell NetWare 6.0 and above. If you find any bugs, or wish to contribute in other ways, please use our bug reporting page.

The bug reporting page and dev-httpd mailing list are *not* provided to answer questions about configuration or running Apache. Before you submit a bug report or request, first consult this document, the Frequently Asked Questions page and the other relevant documentation topics. If you still have a question or problem, post it to the novell.devsup.webserver newsgroup, where many Apache users are more than willing to answer new and obscure questions about using Apache on NetWare.

Most of this document assumes that you are installing Apache from a binary distribution. If you want to compile Apache yourself (possibly to help with development, or to track down bugs), see the section on Compiling Apache for NetWare below.

Requirements

Apache 2.0 is designed to run on NetWare 6.0 service pack 3 and above. If you are running a service pack less than SP3, you must install the latest NetWare Libraries for C (LibC).

NetWare service packs are available here.

Apache 2.0 for NetWare can also be run in a NetWare 5.1 environment as long as the latest service pack or the latest version of the NetWare Libraries for C (LibC) has been installed . **WARNING:** Apache 2.0 for NetWare has not been targeted for or tested in this environment.

Downloading Apache for NetWare

Information on the latest version of Apache can be found on the Apache web server at http://www.apache.org/. This will list the current release, any more recent alpha or beta-test releases, together with details of mirror web and anonymous ftp sites. Binary builds of the latest releases of Apache 2.0 for NetWare can be downloaded from here.

Installing Apache for NetWare

There is no Apache install program for NetWare currently. If you are building Apache 2.0 for NetWare from source, you will need to copy the files over to the server manually.

Follow these steps to install Apache on NetWare from the binary download (assuming you will install to `sys:/apache2`):

- Unzip the binary download file to the root of the `SYS:` volume (may be installed to any volume)
- Edit the `httpd.conf` file setting `ServerRoot` 和 `ServerName` along with any file path values to reflect your correct server settings
- Add `SYS:/APACHE2` to the search path, for example:

```
SEARCH ADD SYS:\APACHE2
```

Follow these steps to install Apache on NetWare manually from your own build source (assuming you will install to `sys:/apache2`):

- Create a directory called `Apache2` on a NetWare volume
- Copy `APACHE2.NLM` , `APRLIB.NLM` to `SYS:/APACHE2`
- Create a directory under `SYS:/APACHE2` called `BIN`
- Copy `HTDIGEST.NLM` , `HTPASSWD.NLM` , `HTDBM.NLM` , `LOGRES.NLM` , `ROTLOGS.NLM` to `SYS:/APACHE2/BIN`
- Create a directory under `SYS:/APACHE2` called `CONF`
- Copy the `HTTPD-STD.CONF` file to the `SYS:/APACHE2/CONF` directory and rename to `HTTPD.CONF`
- Copy the `MIME.TYPES` , `CHARSET.CONV` 和 `MAGIC` files to `SYS:/APACHE2/CONF` directory
- Copy all files and subdirectories in `\HTTPD-2.0\DOCS\ICONS` to `SYS:/APACHE2/ICONS`
- Copy all files and subdirectories in `\HTTPD-2.0\DOCS\MANUAL` to `SYS:/APACHE2/MANUAL`
- Copy all files and subdirectories in `\HTTPD-2.0\DOCS\ERROR` to `SYS:/APACHE2/ERROR`
- Copy all files and subdirectories in `\HTTPD-2.0\DOCS\DOCROOT` to `SYS:/APACHE2/HTDOCS`
- Create the directory `SYS:/APACHE2/LOGS` on the server
- Create the directory `SYS:/APACHE2/CGI-BIN` on the server
- Create the directory `SYS:/APACHE2/MODULES` and copy all nlm modules into the `modules` directory
- Edit the `HTTPD.CONF` file searching for all `@@Value@@` markers and replacing them with the appropriate setting
- Add `SYS:/APACHE2` to the search path, for example:

```
SEARCH ADD SYS:\APACHE2
```

Apache may be installed to other volumes besides the default `SYS` volume.

During the build process, adding the keyword "install" to the makefile command line will automatically produce a complete distribution package under the subdirectory `DIST` . Install Apache by simply copying the distribution that was produced by the makfiles to the root of a

NetWare volume (see: Compiling Apache for NetWare below).

Running Apache for NetWare

To start Apache just type `apache` at the console. This will load apache in the OS address space. If you prefer to load Apache in a protected address space you may specify the address space with the load statement as follows:

```
load address space = apache2 apache2
```

This will load Apache into an address space called apache2. Running multiple instances of Apache concurrently on NetWare is possible by loading each instance into its own protected address space.

After starting Apache, it will be listening to port 80 (unless you changed the `Listen` directive in the configuration files). To connect to the server and access the default page, launch a browser and enter the server's name or address. This should respond with a welcome page, and a link to the Apache manual. If nothing happens or you get an error, look in the `error_log` file in the `logs` directory.

Once your basic installation is working, you should configure it properly by editing the files in the `conf` directory.

To unload Apache running in the OS address space just type the following at the console:

```
unload apache2
```

或

```
apache2 shutdown
```

If apache is running in a protected address space specify the address space in the unload statement:

```
unload address space = apache2 apache2
```

When working with Apache it is important to know how it will find the configuration files. You can specify a configuration file on the command line in two ways:

- `-f` specifies a path to a particular configuration file

```
apache2 -f "vol:/my server/conf/my.conf"
```

```
apache -f test/test.conf
```

In these cases, the proper `ServerRoot` should be set in the configuration file.

If you don't specify a configuration file name with `-f`, Apache will use the file name compiled into the server, usually `conf/httpd.conf`. Invoking Apache with the `-V` switch will display this value labeled as `SERVER_CONFIG_FILE`. Apache will then determine its `ServerRoot` by trying the following, in this order:

- A `ServerRoot` directive via a `-C` switch.
- The `-d` switch on the command line.
- Current working directory
- The server root compiled into the server.

The server root compiled into the server is usually `sys:/apache2`. invoking apache with the `-V` switch will display this value labeled as `HTTPD_ROOT`.

Apache 2.0 for NetWare includes a set of command line directives that can be used to modify or display information about the running instance of the web server. These directives are only available while Apache is running. Each of these directives must be preceded by the keyword `APACHE2`.

RESTART

Instructs Apache to terminate all running worker threads as they become idle, reread the configuration file and restart each worker thread based on the new configuration.

VERSION

Displays version information about the currently running instance of Apache.

MODULES

Displays a list of loaded modules both built-in and external.

DIRECTIVES

Displays a list of all available directives.

SETTINGS

Enables or disables the thread status display on the console. When enabled, the state of each running threads is displayed on the Apache console screen.

SHUTDOWN

Terminates the running instance of the Apache web server.

HELP

Describes each of the runtime directives.

By default these directives are issued against the instance of Apache running in the OS address space. To issue a directive against a specific instance running in a protected address space, include the -p parameter along with the name of the address space. For more information type "apache2 Help" on the command line.

Configuring Apache for NetWare

Apache is configured by reading configuration files usually stored in the `conf` directory. These are the same as files used to configure the Unix version, but there are a few different directives for Apache on NetWare. See the Apache documentation for all the available directives.

The main differences in Apache for NetWare are:

- Because Apache for NetWare is multithreaded, it does not use a separate process for each request, as Apache does on some Unix implementations. Instead there are only threads running: a parent thread, and multiple child or worker threads which handle the requests.

  Therefore the "process"-management directives are different:

  `MaxRequestsPerChild` - Like the Unix directive, this controls how many requests a worker thread will serve before exiting. The recommended default, `MaxRequestsPerChild 0`, causes the thread to continue servicing request indefinitely. It is recommended on NetWare, unless there is some specific reason, that this directive always remain set to `0`.

  `StartThreads` - This directive tells the server how many threads it should start initially. The recommended default is `StartThreads 50`.

  `MinSpareThreads` - This directive instructs the server to spawn additional worker threads if the number of idle threads ever falls below this value. The recommended default is `MinSpareThreads 10`.

  `MaxSpareThreads` - This directive instructs the server to begin terminating worker threads if the number of idle threads ever exceeds this value. The recommended default is `MaxSpareThreads 100`.

  `MaxThreads` - This directive limits the total number of work threads to a maximum value. The recommended default is `ThreadsPerChild 250`.

`ThreadStackSize` - This directive tells the server what size of stack to use for the individual worker thread. The recommended default is `ThreadStackSize 65536`.

- The directives that accept filenames as arguments must use NetWare filenames instead of Unix names. However, because Apache uses Unix-style names internally, forward slashes must be used rather than backslashes. It is recommended that all rooted file paths begin with a volume name. If omitted, Apache will assume the `SYS:` volume which may not be correct.
- Apache for NetWare has the ability to load modules at runtime, without recompiling the server. If Apache is compiled normally, it will install a number of optional modules in the `\Apache2\modules` directory. To activate these, or other modules, the `LoadModule` directive must be used. For example, to active the status module, use the following:

```
LoadModule status_module modules/status.nlm
```

Information on creating loadable modules is also available.

Additional NetWare specific directives:

- `CGIMapExtension` - This directive maps a CGI file extension to a script interpreter.
- `SecureListen` - Enables SSL encryption for a specified port.
- `NWSSLTrustedCerts` - Adds trusted certificates that are used to create secure connections to proxied servers.
- `NWSSLUpgradeable` - Allow a connection created on the specified address/port to be upgraded to an SSL connection.

Compiling Apache for NetWare

Compiling Apache requires MetroWerks CodeWarrior 6.x or higher. Once Apache has been built, it can be installed to the root of any NetWare volume. The default is the `sys:/Apache2` directory.

Before running the server you must fill out the `conf` directory. Copy the file `HTTPD-STD.CONF` from the distribution `conf` directory and rename it to `HTTPD.CONF`. Edit the `HTTPD.CONF` file searching for all `@@Value@@` markers and replacing them with the appropriate setting. Copy over the `conf/magic` 和 `conf/mime.types` files as well. Alternatively, a complete distribution can be built by including the keyword `install` when invoking the makefiles.

Requirements:

The following development tools are required to build Apache 2.0 for NetWare:

- Metrowerks CodeWarrior 6.0 or higher with the NetWare PDK 3.0 or higher.
- NetWare Libraries for C (LibC)
- LDAP Libraries for C
- ZLIB Compression Library source code
- AWK utility (awk, gawk or similar). AWK can be downloaded from http://developer.novell.com/ndk/apache.htm. The utility must be found in your windows path and must be named `awk.exe`.
- To build using the makefiles, you will need GNU make version 3.78.1 (GMake) available at http://developer.novell.com/ndk/apache.htm.

Building Apache using the NetWare makefiles:

- Set the environment variable `NOVELLLIBC` to the location of the NetWare Libraries for C SDK, for example:

  ```
  Set NOVELLLIBC=c:\novell\ndk\libc
  ```

- Set the environment variable `METROWERKS` to the location where you installed the Metrowerks CodeWarrior compiler, for example:

  ```
  Set METROWERKS=C:\Program Files\Metrowerks\CodeWarrior
  ```

  If you installed to the default location `C:\Program Files\Metrowerks\CodeWarrior`, you don't need to set this.

- Set the environment variable `LDAPSDK` to the location where you installed the LDAP Libraries for C, for example:

  ```
  Set LDAPSDK=c:\Novell\NDK\cldapsdk\NetWare\libc
  ```

- Set the environment variable `ZLIBSDK` to the location where you installed the source code for the ZLib Library, for example:

  ```
  Set ZLIBSDK=D:\NOVELL\zlib
  ```

- Set the environment variable `AP_WORK` to the full path of the `httpd` source code directory.

  ```
  Set AP_WORK=D:\httpd-2.0.x
  ```

- Set the environment variable `APR_WORK` to the full path of the `apr` source code directory. Typically `\httpd\srclib\apr` but the APR project can be outside of the httpd directory structure.

```
Set APR_WORK=D:\apr-1.x.x
```

- Set the environment variable `APU_WORK` to the full path of the `apr-util` source code directory. Typically `\httpd\srclib\apr-util` but the APR-UTIL project can be outside of the httpd directory structure.

```
Set APU_WORK=D:\apr-util-1.x.x
```

- Make sure that the path to the AWK utility and the GNU make utility (`gmake.exe`) have been included in the system's `PATH` environment variable.

- Download the source code and unzip to an appropriate directory on your workstation.
- Change directory to `\httpd-2.0` and build the prebuild utilities by running " `gmake -f nwgnumakefile prebuild` ". This target will create the directory `\httpd-2.0\nwprebuild` and copy each of the utilities to this location that are necessary to complete the following build steps.
- Copy the files `\httpd-2.0\nwprebuild\GENCHARS.nlm` 和 `\httpd-2.0\nwprebuild\DFTABLES.nlm` to the `SYS:` volume of a NetWare server and run them using the following commands:

```
            SYS:\genchars &gt; sys:\test_char.h

            SYS:\dftables sys:\chartables.c
```

- Copy the files `test_char.h` 和 `chartables.c` to the directory `\httpd-2.0\os\netware` on the build machine.

- Change directory to `\httpd-2.0` and build Apache by running " `gmake -f nwgnumakefile` ". You can create a distribution directory by adding an install parameter to the command, for example:

```
gmake -f nwgnumakefile install
```

Additional make options

- `gmake -f nwgnumakefile`

  Builds release versions of all of the binaries and copies them to a `\release` destination directory.

- `gmake -f nwgnumakefile DEBUG=1`

  Builds debug versions of all of the binaries and copies them to a `\debug` destination directory.

- `gmake -f nwgnumakefile install`

  Creates a complete Apache distribution with binaries, docs and additional support files in a `\dist\Apache2` directory.

- `gmake -f nwgnumakefile prebuild`

  Builds all of the prebuild utilities and copies them to the `\nwprebuild` directory.

- `gmake -f nwgnumakefile installdev`

  Same as install but also creates a `\lib` 和 `\include` directory in the destination directory and copies headers and import files.

- `gmake -f nwgnumakefile clean`

  Cleans all object files and binaries from the `\release.o` 或 `\debug.o` build areas depending on whether `DEBUG` has been defined.

- `gmake -f nwgnumakefile clobber_all`

  Same as clean and also deletes the distribution directory if it exists.

Additional environment variable options

- To build all of the experimental modules, set the environment variable `EXPERIMENTAL` :

  ```
  Set EXPERIMENTAL=1
  ```

- To build Apache using standard BSD style sockets rather than Winsock, set the environment variable `USE_STDSOCKETS` :

  ```
  Set USE_STDSOCKETS=1
  ```

Building mod_ssl for the NetWare platform

By default Apache for NetWare uses the built-in module `mod_nw_ssl` to provide SSL services. This module simply enables the native SSL services implemented in NetWare OS to handle all encryption for a given port. Alternatively, mod_ssl can also be used in the same manner as on other platforms.

Before mod_ssl can be built for the NetWare platform, the OpenSSL libraries must be provided. This can be done through the following steps:

- Download the latest NetWare patch for OpenSSL from the OpenSSL Contribution page.
- Download the corresponding OpenSSL source code from the OpenSSL Source page.
- At the root of the OpenSSL source directory, apply the NetWare patch using the "patch" utility, for example:

```
patch -p 1 -i netwarepatch-0.9.7g.diff
```

- Edit the file `NetWare/set_env.bat` and modify any tools and utilities paths so that they correspond to your build environment.
- From the root of the OpenSSL source directory, run the following scripts:

```
              Netware/set_env netware-libc

              Netware/build netware-libc
```

- Before building Apache, set the environment variable `OSSLSDK` to the full path to the root of the openssl source code directory.

```
Set OSSLSDK=d:\openssl-0.9.7x
```

Running a High-Performance Web Server on HPUX

```
Date: Wed, 05 Nov 1997 16:59:34 -0800
From: Rick Jones <[raj@cup.hp.com](mailto:raj@cup.hp.com)>
Reply-To: [raj@cup.hp.com](mailto:raj@cup.hp.com)
Organization: Network Performance
Subject: HP-UX tuning tips
```

Here are some tuning tips for HP-UX to add to the tuning page.

For HP-UX 9.X: Upgrade to 10.20 For HP-UX 10.[00|01|10]: Upgrade to 10.20

For HP-UX 10.20:

Install the latest cumulative ARPA Transport Patch. This will allow you to configure the size of the TCP connection lookup hash table. The default is 256 buckets and must be set to a power of two. This is accomplished with adb against the *disc* image of the kernel. The variable name is `tcp_hash_size`. Notice that it's critically important that you use "`W`" to write a 32 bit quantity, not "`w`" to write a 16 bit value when patching the disc image because the `tcp_hash_size` variable is a 32 bit quantity.

How to pick the value? Examine the output of ftp://ftp.cup.hp.com/dist/networking/tools/connhist and see how many total TCP connections exist on the system. You probably want that number divided by the hash table size to be reasonably small, say less than 10. Folks can look at HP's SPECweb96 disclosures for some common settings. These can be found at http://www.specbench.org/. If an HP-UX system was performing at 1000 SPECweb96 connections per second, the `TIME_WAIT` time of 60 seconds would mean 60,000 TCP "connections" being tracked.

Folks can check their listen queue depths with ftp://ftp.cup.hp.com/dist/networking/misc/listenq.

If folks are running Apache on a PA-8000 based system, they should consider "chatr'ing" the Apache executable to have a large page size. This would be "`chatr +pi L <BINARY>`". The GID of the running executable must have `MLOCK` privileges. `Setprivgrp(1m)` should be consulted for assigning `MLOCK`. The change can be validated by running Glance and examining the memory regions of the server(s) to make sure that they show a non-trivial fraction of the text segment being locked.

If folks are running Apache on MP systems, they might consider writing a small program that uses `mpctl()` to bind processes to processors. A simple `pid % numcpu` algorithm is probably sufficient. This might even go into the source code.

If folks are concerned about the number of `FIN_WAIT_2` connections, they can use nettune to shrink the value of `tcp_keepstart`. However, they should be careful there - certainly do not make it less than oh two to four minutes. If `tcp_hash_size` has been set well, it is probably OK to let the `FIN_WAIT_2`'s take longer to timeout (perhaps even the default two hours) - they will not on average have a big impact on performance.

There are other things that could go into the code base, but that might be left for another email. Feel free to drop me a message if you or others are interested.

sincerely,

rick jones

http://www.cup.hp.com/netperf/NetperfPage.html

The Apache EBCDIC Port

Warning: This document has not been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

Overview of the Apache EBCDIC Port

Version 1.3 of the Apache HTTP Server is the first version which includes a port to a (non-ASCII) mainframe machine which uses the EBCDIC character set as its native codeset.

(It is the SIEMENS family of mainframes running the BS2000/OSD operating system. This mainframe OS nowadays features a SVR4-derived POSIX subsystem).

The port was started initially to

- prove the feasibility of porting the Apache HTTP server to this platform
- find a "worthy and capable" successor for the venerable CERN-3.0 daemon (which was ported a couple of years ago), and to
- prove that Apache's preforking process model can on this platform easily outperform the accept-fork-serve model used by CERN by a factor of 5 or more.

This document serves as a rationale to describe some of the design decisions of the port to this machine.

Design Goals

One objective of the EBCDIC port was to maintain enough backwards compatibility with the (EBCDIC) CERN server to make the transition to the new server attractive and easy. This required the addition of a configurable method to define whether a HTML document was stored in ASCII (the only format accepted by the old server) or in EBCDIC (the native document format in the POSIX subsystem, and therefore the only realistic format in which the other POSIX tools like `grep` 或 `sed` could operate on the documents). The current solution to this is a "pseudo-MIME-format" which is intercepted and interpreted by the Apache server (see below). Future versions might solve the problem by defining an "ebcdic-handler" for all documents which must be converted.

Technical Solution

Since all Apache input and output is based upon the BUFF data type and its methods, the easiest solution was to add the conversion to the BUFF handling routines. The conversion must be settable at any time, so a BUFF flag was added which defines whether a BUFF object has currently enabled conversion or not. This flag is modified at several points in the HTTP protocol:

- **set** before a request is received (because the request and the request header lines are always in ASCII format)
- **set/unset** when the request body is received - depending on the content type of the request body (because the request body may contain ASCII text or a binary file)
- **set** before a reply header is sent (because the response header lines are always in ASCII format)
- **set/unset** when the response body is sent - depending on the content type of the response body (because the response body may contain text or a binary file)

Porting Notes

1. The relevant changes in the source are `#ifdef` 'ed into two categories:

   `**#ifdef CHARSET_EBCDIC**`

   Code which is needed for any EBCDIC based machine. This includes character translations, differences in contiguity of the two character sets, flags which indicate which part of the HTTP protocol has to be converted and which part doesn't *etc.*

   `**#ifdef _OSD_POSIX**`

   Code which is needed for the SIEMENS BS2000/OSD mainframe platform only. This deals with include file differences and socket implementation topics which are only required on the BS2000/OSD platform.

2. The possibility to translate between ASCII and EBCDIC at the socket level (on BS2000 POSIX, there is a socket option which supports this) was intentionally *not* chosen, because the byte stream at the HTTP protocol level consists of a mixture of protocol related strings and non-protocol related raw file data. HTTP protocol strings are always encoded in ASCII (the `GET` request, any Header: lines, the chunking information *etc.*) whereas the file transfer parts (*i.e.*, GIF images, CGI output *etc.*) should usually be just "passed through" by the server. This separation between "protocol string" and "raw data" is reflected in the server code by functions like `bgets()` 或 `rvputs()` for strings, and functions like `bwrite()` for binary data. A global translation of everything would therefore be inadequate.

   (In the case of text files of course, provisions must be made so that EBCDIC documents are always served in ASCII)

3. This port therefore features a built-in protocol level conversion for the server-internal strings (which the compiler translated to EBCDIC strings) and thus for all server-generated documents. The hard coded ASCII escapes `\012` 和 `\015` which are ubiquitous in the server code are an exception: they are already the binary encoding of the ASCII `\n` 和 `\r` and must not be converted to ASCII a second time. This exception is only relevant for server-generated strings; and *external* EBCDIC documents are not expected to contain ASCII newline characters.

4. By examining the call hierarchy for the BUFF management routines, I added an "ebcdic/ascii conversion layer" which would be crossed on every puts/write/get/gets, and a conversion flag which allowed enabling/disabling the conversions on-the-fly. Usually, a document crosses this layer twice from its origin source (a file or CGI output) to its destination (the requesting client): `file -> Apache`, and `Apache -> client`.

   The server can now read the header lines of a CGI-script output in EBCDIC format, and then find out that the remainder of the script's output is in ASCII (like in the case of the output of a WWW Counter program: the document body contains a GIF image). All header processing is done in the native EBCDIC format; the server then determines, based on the type of document being served, whether the document body (except for the chunking information, of course) is in ASCII already or must be converted from EBCDIC.

5. For Text documents (MIME types text/plain, text/html *etc.*), an implicit translation to ASCII can be used, or (if the users prefer to store some documents in raw ASCII form for faster serving, or because the files reside on a NFS-mounted directory tree) can be served without conversion.

   Example:

   to serve files with the suffix `.ahtml` as a raw ASCII `text/html` document without implicit conversion (and suffix `.ascii` as ASCII `text/plain`), use the directives:

   ```
   AddType  text/x-ascii-html  .ahtml
   AddType  text/x-ascii-plain .ascii
   ```

   Similarly, any `text/foo` MIME type can be served as "raw ASCII" by configuring a MIME type "`text/x-ascii-foo`" for it using `AddType`.

6. Non-text documents are always served "binary" without conversion. This seems to be the most sensible choice for, .例如，GIF/ZIP/AU file types. This of course requires the user to copy them to the mainframe host using the "`rcp -b`" binary switch.

7. Server parsed files are always assumed to be in native (*i.e.*, EBCDIC) format as used on the machine, and are converted after processing.
8. For CGI output, the CGI script determines whether a conversion is needed or not: by setting the appropriate Content-Type, text files can be converted, or GIF output can be passed through unmodified. An example for the latter case is the wwwcount program which we ported as well.

Document Storage Notes

Binary Files

All files with a `Content-Type:` which does not start with `text/` are regarded as *binary files* by the server and are not subject to any conversion. Examples for binary files are GIF images, gzip-compressed files and the like.

When exchanging binary files between the mainframe host and a Unix machine or Windows PC, be sure to use the ftp "binary" (`TYPE I`) command, or use the `rcp -b` command from the mainframe host (the `-b` switch is not supported in unix `rcp` 's).

Text Documents

The default assumption of the server is that Text Files (*i.e.*, all files whose `Content-Type:` starts with `text/`) are stored in the native character set of the host, EBCDIC.

Server Side Included Documents

SSI documents must currently be stored in EBCDIC only. No provision is made to convert it from ASCII before processing.

Apache Modules' Status

Module	Status	Notes
`core`	+	
`mod_authz_host`	+	
`mod_actions`	+	
`mod_alias`	+	
`mod_asis`	+	
`mod_auth_basic`	+	

mod_authn_file	+	
mod_authn_anon	+	
mod_authn_dbm	?	with own libdb.a
mod_autoindex	+	
mod_cern_meta	?	
mod_cgi	+	
mod_digest	+	
mod_dir	+	
mod_so	-	no shared libs
mod_env	+	
mod_example	-	(test bed only)
mod_expires	+	
mod_headers	+	
mod_imagemap	+	
mod_include	+	
mod_info	+	
mod_log_agent	+	
mod_log_config	+	
mod_mime	+	
mod_mime_magic	?	not ported yet
mod_negotiation	+	
mod_proxy	+	
mod_rewrite	+	untested
mod_setenvif	+	
mod_speling	+	
mod_status	+	
mod_unique_id	+	
mod_userdir	+	
mod_usertrack	?	untested

Third Party Modules' Status

Module	Status	Notes
`mod_jserv`	-	JAVA still being ported.
`mod_php3`	+	`mod_php3` runs fine, with LDAP and GD and FreeType libraries.
`mod_put`	?	untested
`mod_session`	-	untested

服务器和支持程序

本文列出了Apache HTTP服务器中所有的可执行程序。

索引

`httpd`

Apache超文本传输协议服务器

`apachectl`

Apache HTTP服务器控制接口

`ab`

Apache HTTP服务器性能测试工具

`apxs`

APache功能扩展工具

`configure`

配置源代码树

`dbmmanage`

建立和更新DBM形式的基本认证文件

`htcacheclean`

清理磁盘缓冲区

`htdigest`

建立和更新摘要认证文件

`htdbm`

操作DBM数据库文件

`htpasswd`

建立和更新基本认证文件

`httxt2dbm`

创建RewriteMap指令需要使用的dbm文件

`logresolve`

将Apache日志文件中的IP地址解析为主机名

`rotatelogs`

滚动Apache日志而无须终止服务器

`suexec`

为Exec切换用户

其他程序

没有单独手册页面的支持程序

httpd - Apache超文本传输协议服务器

`httpd` 是Apache超文本传输协议(HTTP)服务器的主程序。被设计为一个独立运行的后台进程，它会建立一个处理请求的子进程或线程的池。

通常， `httpd` 不应该被直接调用，而应该在类Unix系统中由 `apachectl` 调用，在Windows NT/2000/XP/2003中作为服务运行和在Windows 95/98/ME中作为控制台程序运行.

语法

```
**httpd** [ -**d** serverroot ] [ -**f** config ] [ -**C** directive ] [ -**c** directive
```

在中Windows系统，还可以使用下列参数：

```
**httpd** [ -**k** install|config|uninstall ] [ -**n** name ] [ -**w** ]
```

选项

`-d serverroot`

将 `ServerRoot` 指令设置初始值为serverroot。它可以被配置文件中的 `ServerRoot` 指令所覆盖。其默认值是 `/usr/local/apache2` 。

`-f config`

在启动中使用config作为配置文件。如果config不以"/"开头，则它是相对于 `ServerRoot` 的路径。其默认值是 `conf/httpd.conf` 。

`-k` `start|restart|graceful|stop|graceful-stop`

发送信号使 `httpd` 启动、重新启动或停止。更多信息请参见停止Apache 。

`-C directive`

在读取配置文件之前，先处理directive的配置指令。

`-c directive`

在读取配置文件之后，再处理directive的配置指令。

`-D parameter`

设置参数parameter， 它配合配置文件中的 `<IfDefine>` 段，用于在服务器启动和重新启动时，有条件地跳过或处理某些命令。

`-e level`

在服务器启动时，设置 `LogLevel` 为level 。它用于在启动时，临时增加出错信息的详细程度，以帮助排错。

`-E file`

将服务器启动过程中的出错信息发送到文件file 。

`-R directory`

当在服务器编译中使用了 `SHARED_CORE` 规则时，它指定共享目标文件的目录为directory 。

`-h`

输出一个可用的命令行选项的简要说明。

`-l`

输出一个静态编译在服务器中的模块的列表。它不会列出使用 `LoadModule` 指令动态加载的模块。

`-L`

输出一个指令的列表，并包含了各指令的有效参数和使用区域。

`-M`

输出一个已经启用的模块列表，包括静态编译在服务器中的模块和作为DSO动态加载的模块。

`-S`

显示从配置文件中读取并解析的设置结果(目前仅显示虚拟主机的设置)

`-t`

仅对配置文件执行语法检查。程序在语法解析检查结束后立即退出，或者返回"0"(OK)，或者返回非0的值(Error)。如果还指定了"-D DUMP_VHOSTS"，则会显示虚拟主机配置的详细信息。

`-v`

显示 `httpd` 的版本，然后退出。

`-V`

显示 `httpd` 的版本和编译参数，然后退出。

`-X`

以调试模式运行 `httpd` 。仅启动一个工作进程，并且服务器不与控制台脱离。

下列参数仅用于Windows平台 ：

`-k install|config|uninstall`

安装Apache为一个Windows NT的服务；改变Apache服务的启动方式；删除Apache服务。

`-n name`

指定Apache服务的名称为name

`-w`

保持打开控制台窗口，使得可以阅读出错信息。

ab - Apache HTTP服务器性能测试工具

`ab` 是Apache超文本传输协议(HTTP)的性能测试工具。其设计意图是描绘当前所安装的Apache的执行性能，主要是显示你安装的Apache每秒可以处理多少个请求。

语法

```
**ab** [ -**A** auth-username:password ] [ -**c** concurrency ] [ -**C** cookie-name=value
```

选项

`-A auth-username:password`

向服务器提供基本认证信息。用户名和密码之间由一个" `:` "隔开，并将被以base64编码形式发送。无论服务器是否需要(即是否发送了401认证需求代码)，此字符串都会被发送。

`-c concurrency`

一次产生的请求个数。默认是一次一个。

`-C cookie-name=value`

对请求附加一个" `Cookie:` "头行。其典型形式是 `name=value` 的一个参数对。此参数可以重复。

`-d`

不显示"percentage served within XX [ms] table"消息(为以前的版本提供支持)。

`-e csv-file`

产生一个逗号分隔(CSV)文件，其中包含了处理每个相应百分比请求(从1%到100%)所需要的相应百分比时间(以微秒为单位)。由于这种格式已经"二进制化"，所以比"gnuplot"格式更有用。

`-g gnuplot-file`

把所有测试结果写入一个"gnuplot"或者TSV(以Tab分隔)文件。此文件可以方便地导入到Gnuplot, IDL, Mathematica, Excel中。其中的第一行为标题。

`-h`

显示使用方法的帮助信息。

`-H custom-header`

对请求附加额外的头信息。此参数的典型形式是一个有效的头信息行，其中包含了以冒号分隔的字段和值(如： `"Accept-Encoding: zip/zop;8bit"`)。

`-i`

执行 `HEAD` 请求，而不是 `GET` 。

`-k`

启用KeepAlive功能，即在一个HTTP会话中执行多个请求。默认不启用KeepAlive功能。

`-n requests`

在测试会话中所执行的请求个数。默认仅执行一个请求，此时其结果不具有意义。

`-p POST-file`

包含了POST数据的文件。

`-P proxy-auth-username:password`

对一个中转代理提供基本认证信息。用户名和密码由一个" `:` "隔开，并将被以base64编码形式发送。无论服务器是否需要(即是否发送了407代理认证需求代码)，此字符串都会被发送。

`-q`

如果处理的请求数大于150， `ab` 每处理大约10%或者100个请求时，会在 `stderr` 输出一个进度计数。此 `-q` 标记可以屏蔽这些信息。

`-s`

用于编译中(`ab -h` 会告诉你)使用了SSL的受保护的 `https` ，而不是 `http` 协议的时候。此功能是实验性的，最好不要用。

`-S`

不显示中值和标准偏差值，而且在均值和中值为标准偏差值的1到2倍时，也不显示警告或出错信息。默认时，会显示最小值/均值/最大值等数值。(为以前的版本提供支持)

`-t timelimit`

测试所进行的最大秒数。内部隐含值是" `-n 50000` "。它可以使对服务器的测试限制在一个固定的总时间以内。默认时，没有时间限制。

`-T content-type`

POST数据时所使用的"Content-type"头信息。

`-v verbosity`

设置显示信息的详细程度， `4` 或更大值会显示头信息， `3` 或更大值可以显示响应代码(404，200等)， `2` 或更大值可以显示警告和其他信息。

`-V`

显示版本号并退出。

`-w`

以HTML表格形式输出结果。默认时，它是白色背景的两列宽度的一张表。

`-x <table>-attributes`

设置 `<table>` 属性的字符串。此属性被填入 `<table 这里 >` 。

`-X proxy[:port]`

对请求使用代理服务器。

`-y <tr>-attributes`

设置 `<tr>` 属性的字符串。

`-z <td>-attributes`

设置 `<td>` 属性的字符串。

Bugs

程序中有各种静态声明的固定长度的缓冲区。另外，对命令行参数、服务器的响应头和其他外部输入的解析也很简单，这可能会有不良后果。

没有完整实现HTTP/1.x ；仅接受某些"预想"的响应格式。 `strstr()` 的频繁使用可能会带来性能问题，即你可能是在测试 `ab` 而不是服务器的性能。

apachectl - Apache HTTP服务器控制接口

`apachectl` 是Apache HTTP服务器的前端程序。其设计意图是帮助管理员控制Apache `httpd` 后台守护进程的功能。

`apachectl` 脚本有两种操作模式。首先，作为简单的 `httpd` 的前端程序，设置所有必要的环境变量，然后启动 `httpd` ，并传递所有的命令行参数。其次，作为SysV初始化脚本，接受简单的一个单词的参数，如：`start`，`restart`，`stop` ，并把他们翻译为适当的信号发送给 `httpd` 。

如果你的Apache安装在非标准的路径中，你将需要修改 `apachectl` 脚本使其中的路径正确地指向 `httpd` 程序。此外，还可以指定任何必要的 `httpd` 命令行参数。细节可以参见脚本中的注解。

`apachectl` 脚本如果执行成功，则返回0 ；如果出错，则其返回值>0 。更多细节可以参见脚本中的注解。

语法

在扮演传递角色时， `apachectl` 可以接受对 `httpd` 程序有效的所有参数。

`**apachectl** [ httpd-argument ]`

在SysV初始化模式中， `apachectl` 只接受简单的一个单词的命令，如下：

`**apachectl** command`

选项

下列仅说明了SysV初始化类型的选项，其他参数的说明见 `httpd` 手册页。

`start`

启动Apache `httpd` 后台守护进程。如果已经启动，则产生一个错误。它等价于 `apachectl -k start` 。

`stop`

停止Apache `httpd` 后台守护进程。它等价于 `apachectl -k stop` 。

`restart`

重新启动Apache `httpd` 后台守护进程。如果守护进程尚未运行，则启动它。在重新启动守护进程之前，此命令会使用 `configtest` 自动检查配置文件，以确保Apache不会死掉。它等价于 `apachectl -k restart` 。

`fullstatus`

显示由 `mod_status` 提供的完整的状态报告。要使用这个功能，需要启用服务器上的 `mod_status` 模块，并且系统中有一个基于文本的浏览器，如 `lynx` 。修改脚本中的 `STATUSURL` 变量，可以修改访问状态报告的URL 。

`status`

显示一个简要的状态报告。它类似于 `fullstatus` 选项，但是省略了正在处理的请求的列表。

`graceful`

优雅地重新启动Apache `httpd` 后台守护进程。如果守护进程尚未启动，则启动它。它和标准重新启动的不同之处在于：不会中断当前已经打开的连接，也不会立即关闭日志。这意味着，如果在日志滚动脚本使用它，则在处理日志之前必须加入一个实实在在的延迟，以确保老的日志文件在被处理前已经关闭。在重新启动守护进程之前，此命令会使用 `configtest` 自动检查配置文件，以确保Apache不会死掉。它等价于 `apachectl -k graceful` 。

`graceful-stop`

优雅地停止Apache `httpd` 后台守护进程。它和标准停止的不同之处在于：不会中断当前已经打开的连接，也不会立即关闭日志。它等价于 `apachectl -k graceful-stop` 。

`configtest`

执行一次配置文件语法检查。它解析配置文件，并报告 `Syntax Ok` 或者是特定的语法错误详细信息。它等价于 `apachectl -t` 。

下列选项仅在早期版本中使用，现在已经被废弃了。

`startssl`

以支持SSL的方式启动 `httpd` ，你应当编辑配置文件，并在其中包含与SSL支持相关的指令，然后使用 `apachectl start` 启动服务器。

apxs - Apache 扩展工具

`apxs` 是一个为Apache HTTP服务器编译和安装扩展模块的工具，用于编译一个或多个源程序或目标代码文件为动态共享对象，使之可以用由 `mod_so` 提供的 `LoadModule` 指令在运行时加载到Apache服务器中。

因此，要使用这个扩展机制，你的平台必须支持DSO特性，而且Apache `httpd` 必须内建了 `mod_so` 模块。 `apxs` 工具能自动探测是否具备这样的条件，你也可以自己用这个命令手动探测：

```
$ httpd -l
```

该命令的输出列表中应该有 `mod_so` 模块。如果所有这些条件均已具备，则可以很容易地借助 `apxs` 安装你自己的DSO模块以扩展Apache服务器的功能：

```
$ apxs -i -a -c mod_foo.c
gcc -fpic -DSHARED_MODULE -I/path/to/apache/include -c mod_foo.c
ld -Bshareable -o mod_foo.so mod_foo.o
cp mod_foo.so /path/to/apache/modules/mod_foo.so
chmod 755 /path/to/apache/modules/mod_foo.so
[activating module 'foo' in /path/to/apache/etc/httpd.conf]
$ apachectl restart
/path/to/apache/sbin/apachectl restart: httpd not running, trying to start
[Tue Mar 31 11:27:55 1998] [debug] mod_so.c(303): loaded module foo_module
/path/to/apache/sbin/apachectl restart: httpd started
$ _
```

其中的参数files可以是任何C源程序文件(.c)、目标代码文件(.o)、甚至是一个库(.a)。 `apxs` 工具会根据其后缀自动编译C源程序或者连接目标代码和库。但是，使用预编译的目标代码时，必须保证它们是地址独立代码(PIC)，使之能被动态地加载。如果使用GCC编译，则应该使用 `-fpic` 参数；如果使用其他C编译器，则应该查阅其手册，为 `apxs` 使用相应的编译参数。

有关Apache对DSO的支持的详细信息，可以阅读 `mod_so` 文档，或者直接阅读 `src/modules/standard/mod_so.c` 源程序。

语法

`**apxs** -**g** [ -**S** name=value ] -**n** modname`

```
**apxs** -**q** [ -**S** name=value ] query ...
```

```
**apxs** -**c** [ -**S** name=value ] [ -**o** dsofile ] [ -**I** incdir ] [ -**D** name=va
```

```
**apxs** -**i** [ -**S** name=value ] [ -**n** modname ] [ -**a** ] [ -**A** ] dso-file ..
```

```
**apxs** -**e** [ -**S** name=value ] [ -**n** modname ] [ -**a** ] [ -**A** ] dso-file ..
```

选项

一般选项

`-n modname`

它明确设置了 `-i` (安装)和 `-g` (模板生成)选项的模块名称。对 `-g` 选项，它是必须的；对 `-i` 选项， `apxs` 工具会根据源代码判断，或(在失败的情况下)按文件名推测出这个模块的名称。

查询选项

`-q`

查询某种 `apxs` 设置的信息。该选项的query参数可以是下列一个或多个字符串： `CC` , `CFLAGS` , `CFLAGS_SHLIB` , `INCLUDEDIR` , `LD_SHLIB` , `LDFLAGS_SHLIB` , `LIBEXECDIR` , `LIBS_SHLIB` , `SBINDIR` , `SYSCONFDIR` , `TARGET` 。

这个参数用于手动查询某些设置。比如，要手动处理Apache的C头文件，可以在Makefile中使用：

```
        INC=-I`apxs -q INCLUDEDIR`
```

配置选项

`-S name=value`

此选项可以改变apxs的上述设置。

模板生成选项

`-g`

此选项生成一个名为name的子目录(见选项 `-n`)和其中的两个文件：一个是名为 `mod_name.c` 的样板模块源程序，可以用来建立你自己的模块，或是学习使用apxs机制的良好开端；另一个则是对应的 `Makefile` ，用于编译和安装此模块。

DSO编译选项

`-c`

此选项表示需要执行编译操作。它首先会编译C源程序(.c)files为对应的目标代码文件(.o)，然后连接这些目标代码和files中其余的目标代码文件(.o和.a)，以生成动态共享对象dsofile 。如果没有指定 `-o` 选项，则此输出文件名由files中的第一个文件名推测得到，也就是默认为 `mod_name.so` 。

`-o dsofile`

明确指定所建立的动态共享对象的文件名，它不能从files文件列表中推测得到。如果没有明确指定，则其文件名将为 `mod_unknown.so` 。

`-D name=value`

此选项直接传递到给编译命令，用于增加自定义的编译变量。

`-I incdir`

此选项直接传递到给编译命令，用于增加自定义的包含目录。

`-L libdir`

此选项直接传递到给连接命令，用于增加自定义的库文件目录。

`-l libname`

此选项直接传递到给连接命令，用于增加自定义的库文件。

`-Wc,compiler-flags`

此选项用于向编译命令 `libtool --mode=compile` 中附加compiler-flags ，以增加编译器特有的选项。

`-Wl,linker-flags`

此选项用于向连接命令 `libtool --mode=link` 中附加linker-flags ，以增加连接器特有的选项。

DSO的安装和配置选项

`-i`

此选项表示需要执行安装操作，以安装一个或多个动态共享对象到服务器的modules目录中。

`-a`

此选项自动增加一个 `LoadModule` 行到 `httpd.conf` 文件中，以激活此模块，或者，如果此行已经存在，则启用之。

`-A`

与 `-a` 选项类似，但是它增加的`LoadModule`命令有一个井号前缀(`#`)，即此模块已经准备就绪但尚未启用。

`-e`

表示需要执行编辑操作，它可以与 `-a` 和 `-A` 选项配合使用，与 `-i` 操作类似，修改Apache的`httpd.conf`文件，但是并不安装此模块。

举例

假设有一个扩展Apache功能的模块`mod_foo.c`，使用下列命令，可以将C源程序编译为共享模块，以在运行时加载到Apache服务器中：

```
$ apxs -c mod_foo.c

/path/to/libtool --mode=compile gcc ... -c mod_foo.c

/path/to/libtool --mode=link gcc ... -o mod_foo.la mod_foo.slo

$ _
```

然后，必须修改Apache的配置，以确保有一个`LoadModule`指令来加载此共享对象。为了简化这一步骤，`apxs`可以自动进行该操作，以安装此共享对象到"modules"目录，并更新`httpd.conf`文件，命令如下：

```
$ apxs -i -a mod_foo.la

/path/to/instdso.sh mod_foo.la /path/to/apache/modules

/path/to/libtool --mode=install cp mod_foo.la /path/to/apache/modules
...
chmod 755 /path/to/apache/modules/mod_foo.so

[activating module 'foo' in /path/to/apache/conf/httpd.conf]

$ _
```

如果配置文件中尚不存在，会增加下列的行：

```
LoadModule foo_module modules/mod_foo.so
```

如果你希望默认禁用此模块，可以使用 `-A` 选项，即：

```
$ apxs -i -A mod_foo.c
```

要快速测试apxs机制，可以建立一个Apache模块样板及其对应的Makefile ：

```
$ apxs -g -n foo
Creating [DIR]  foo
Creating [FILE] foo/Makefile
Creating [FILE] foo/modules.mk
Creating [FILE] foo/mod_foo.c
Creating [FILE] foo/.deps
$ _
```

然后，立即可以编译此样板模块为共享对象并加载到Apache服务器中：

```
$ cd foo
$ make all reload
apxs -c mod_foo.c
/path/to/libtool --mode=compile gcc ... -c mod_foo.c
/path/to/libtool --mode=link gcc ... -o mod_foo.la mod_foo.slo
apxs -i -a -n "foo" mod_foo.la
/path/to/instdso.sh mod_foo.la /path/to/apache/modules
/path/to/libtool --mode=install cp mod_foo.la /path/to/apache/modules
...
chmod 755 /path/to/apache/modules/mod_foo.so
[activating module 'foo' in /path/to/apache/conf/httpd.conf]
apachectl restart
/path/to/apache/sbin/apachectl restart: httpd not running, trying to start
[Tue Mar 31 11:27:55 1998] [debug] mod_so.c(303): loaded module foo_module
/path/to/apache/sbin/apachectl restart: httpd started
$ _
```

configure - 配置源代码树

`configure` 脚本配置Apache的源代码树并且将其安装到指定的平台上。丰富的选项允许你根据自己的特定状况和特定需求对Apache进行定制。

这个脚本位于源代码树的根目录下，并且只能用于类Unix操作系统。要了解其他平台的信息，参见：针对特定平台的说明文档。

语法

你必须在源代码树的根目录下调用 `configure` 脚本，语法如下：

```
**./configure** [OPTION]... [VAR=VALUE]...
```

若要指定环境变量(比如：`CC`，`CFLAGS` ...)，请使用 `VAR=VALUE` 的格式。下面有一些有用的环境变量说明。

选项

- 配置选项
- 安装目录
- 系统类型
- 模块选项
- 杂项选项
- 传递给 `apr-config` 脚本的选项
- 特殊程序包选项
- 支持程序选项

配置选项

下列选项会影响 `configure` 脚本自身的行为。方括号"[]"内是默认值。

`-C`

`--config-cache`

等价于 `--cache-file=config.cache`

`--cache-file=FILE`

在FILE文件中缓存测试结果(默认禁用)。

`-h`

`--help [short|recursive]`

显示帮助信息然后退出。使用 `short` 参数将只显示正在运行的当前脚本的选项，而不能列出适用于Apache配置脚本所运行的外部配置脚本的选项。使用 `recursive` 参数将显示所有程序包的简短描述。

`-n`

`--no-create`

`configure` 脚本运行结束后不输出结果文件，常用于正式编译前的测试。

`-q`

`--quiet`

不显示脚本工作期间输出的" `checking ...` "消息。

`--srcdir=DIR`

指定源代码所在目录DIR 。[configure脚本所在目录或父目录]

`--silent`

等价于 `--quiet`

-V

--version

显示版权信息后退出。

安装目录

这些选项控制着安装目录的结构。安装目录的结构取决于所选择的布局。方括号"[]"内是默认值。

`--prefix=PREFIX`

体系无关文件的顶级安装目录PREFIX ， 也就Apache的安装目录。[`/usr/local/apache2`]

`--exec-prefix=EPREFIX`

体系相关文件的顶级安装目录EPREFIX ， 把体系相关的文件安装到不同的位置可以方便地在不同主机之间共享体系相关的文件。[PREFIX]

默认情况下， `make install` 将会把所有文件分别安装到 `/usr/local/apache2/bin` ， `/usr/local/apache2/lib` 目录下。可以用 `--prefix` 指定一个不同于 `/usr/local/apache2` 的安装前缀，比如： `--prefix=$HOME` 。

定义一个目录布局

`--enable-layout=LAYOUT`

使用名为LAYOUT的布局配置所有源代码和编译脚本。这样就允许你在安装Apache时分别指定不同文件的安装位置。 `<a class="pcalibre1 pcalibre">config.layout</a>` 文件中包含了默认布局的示例，你可以根据它创建你自己的布局。这个文件中的不同布局使用 `<Layout FOO>...</Layout>` 段进行分组，其中的 `FOO` 就是布局名。默认的布局是 `Apache` 。

安装目录微调

可以使用下面的选项微调安装目录。下列选项的默认值由 `autoconf` 自动设置并在方括号"[]"内说明。

`--bindir=DIR`

用户可执行目录DIR 。用于存放对网站管理员很有帮助的 `htpasswd` , `dbmmanage` 之类的支持程序。 [`EPREFIX/bin`]

`--datadir=DIR`

Web服务器只读的体系无关数据目录DIR 。虽然 `autoconf` 提供了该选项，但Apache并未使用它。 [`PREFIX/share`]

`--includedir=DIR`

Apache的C头文件目录DIR 。 [`EPREFIX/include`]

`--infodir=DIR`

信息文档目录DIR 。虽然 `autoconf` 提供了该选项，但Apache并未使用它。 [`PREFIX/info`]

`--libdir=DIR`

对象代码库目录DIR 。 [`EPREFIX/lib`]

`--libexecdir=DIR`

程序可执行目录DIR ，也就是动态加载模块目录。 [`EPREFIX/libexec`]

`--localstatedir=DIR`

可写的单一机器数据目录DIR 。虽然 `autoconf` 提供了该选项，但Apache并未使用它。 [`PREFIX/var`]

`--mandir=DIR`

手册文档目录DIR 。 [`EPREFIX/man`]

`--oldincludedir=DIR`

非gcc的C头文件目录DIR 。虽然 `autoconf` 提供了该选项，但Apache并未使用它。 [`/usr/include`]

`--sbindir=DIR`

系统管理员可执行目录DIR，用于存放运行HTTP服务器所必须的 `httpd` ，`apachectl` ，`suexec` 之类的服务程序。 [`EPREFIX/sbin`]

`--sharedstatedir=DIR`

可写的体系无关数据目录DIR 。虽然 `autoconf` 提供了该选项，但Apache并未使用它。[`PREFIX/com`]

`--sysconfdir=DIR`

只读的单一机器数据目录DIR，用于存放 `httpd.conf` 和 `mime.types` 之类的服务器配置文件。[`PREFIX/etc`]

系统类型

这些选项用于交叉编译在其他平台上运行的Apache HTTP服务器。在同一平台上编译和运行Apache HTTP服务器通常不需要使用这些选项，脚本会自动检测并设置。方括号"[]"内是默认值。

`--build=BUILD`

指定编译工具所在系统的系统类型BUILD 。 [`config.guess` 脚本的检测结果]

`--host=HOST`

指定Apache HTTP服务器将要运行的目标系统类型HOST 。 [BUILD]

`--target=TARGET`

configure for building compilers for TARGET类型的系统。虽然 `autoconf` 提供了该选项，但Apache并未使用它。 [HOST]

模块选项

有两种使用模块的方法：一是静态连接进核心，二是作为DSO模块动态加载；如果编译中包含任何DSO模块，则mod_so会被自动包含进核心。如果希望核心能够装载DSO，但不实际编译任何DSO模块，则需明确指定"--enable-so=static"。

一般语法

一般情况下你可以使用如下语法启用或者禁用某个模块：

`--disable-MODULE`

禁用MODULE模块(仅用于基本模块)

`--enable-MODULE=shared`

将MODULE编译为DSO(可用于所有模块)

`--enable-MODULE=static`

将MODULE静态连接进核心(仅用于扩展和实验模块)

`--enable-mods-shared=MODULE-LIST`

将MODULE-LIST中的所有模块都编译成DSO(可用于所有模块)

`--enable-modules=MODULE-LIST`

将MODULE-LIST静态连接进核心(可用于所有模块)

上述 MODULE-LIST 可以是： (1)用引号界定并且用空格分隔的模块名列表

```
            --enable-mods-shared='headers rewrite dav'
```

(2)" `most` "(大多数模块) (3)" `all` "(所有模块)

```
            --enable-mods-shared=most
```

注意

`configure` 将忽略*MODULE*和*MODULE-LIST*中的拼写错误，注意仔细拼写。用于*MODULE*和*MODULE-LIST*中的名称是" `mod_<var class="calibre27">NAME</var>` "中去掉" `mod_` "并将剩余部分中的下划线" `_` "替换为连字符" `-` "以后的结果，比如" `mod_log_config` "模块应当表示为"*log-config*"。

模块列表

基本(B)模块默认包含，必须明确禁用；扩展(E)/实验(X)模块默认不包含，必须明确启用。

| 模块名称 | 状态 | 简要描述 | | `mod_actions` | (B) | 基于媒体类型或请求方法，为执行CGI脚本而提供 | | `mod_alias` | (B) | 提供从文件系统的不同部分到文档树的映射和URL重定向 | | `mod_asis` | (B) | 发送自己包含HTTP头内容的文件 | | `mod_auth_basic` | (B) | 使用基本认证 | | `mod_authn_default` | (B) | 在未正确配置认证模块的情况下简单拒绝一切认证信息 | | `mod_authn_file` | (B) | 使用纯文本文件为认证提供支持 | | `mod_authz_default` | (B) | 在未正确配置授权支持模块的情况下简单拒绝一切授权请求 | | `mod_authz_groupfile` | (B) | 使用纯文本文件为组提供授权支持 | | `mod_authz_host` | (B) | 供基于主机名、IP地址、请求特征的访问控制 | | `mod_authz_user` | (B) | 基于每个用户提供授权支持 | | `mod_autoindex` | (B) | 自动对目录中的内容生成列表，类似于"ls"或"dir"命令 | | `mod_cgi` | (B) | 在非线程型MPM(`prefork`)上提供对CGI脚本执行的支持 | | `mod_cgid` | (B) | 在线程型MPM(`worker`)上用一个外部CGI守护进程执行CGI脚本 | | `mod_dir` | (B) | 指定目录索引文件以及为目录提供"尾斜杠"重定向 | | `mod_env` | (B) | 允许Apache修改或清除传送到CGI脚本和SSI页面的环境变量 | | `mod_filter` | (B) | 根据上下文实际情况对输出过滤器进行动态配置 | | `mod_imagemap` | (B) | 处理服务器端图像映射 | | `mod_include` | (B) | 实现服务端包含文档(SSI)处理 | | `mod_isapi` | (B) | 仅限于在

Windows平台上实现ISAPI扩展 | | `mod_log_config` | (B) | 允许记录日志和定制日志文件格式 | | `mod_mime` | (B) | 根据文件扩展名决定应答的行为(处理器/过滤器)和内容(MIME类型/语言/字符集/编码) | | `mod_negotiation` | (B) | 提供内容协商支持 | | `mod_nw_ssl` | (B) | 仅限于在NetWare平台上实现SSL加密支持 | | `mod_setenvif` | (B) | 根据客户端请求头字段设置环境变量 | | `mod_status` | (B) | 生成描述服务器状态的Web页面 | | `mod_userdir` | (B) | 允许用户从自己的主目录中提供页面(使用"/~username") | | `mod_auth_digest` | (X) | 使用MD5摘要认证(更安全，但是只有最新的浏览器才支持) | | `mod_authn_alias` | (E) | 基于实际认证支持者创建扩展的认证支持者，并为它起一个别名以便于引用 | | `mod_authn_anon` | (E) | 提供匿名用户认证支持 | | `mod_authn_dbd` | (E) | 使用SQL数据库为认证提供支持 | | `mod_authn_dbm` | (E) | 使用DBM数据库为认证提供支持 | | `mod_authnz_ldap` | (E) | 允许使用一个LDAP目录存储用户名和密码数据库来执行基本认证和授权 | | `mod_authz_dbm` | (E) | 使用DBM数据库文件为组提供授权支持 | | `mod_authz_owner` | (E) | 基于文件的所有者进行授权 | | `mod_cache` | (E) | 基于URI键的内容动态缓冲(内存或磁盘) | | `mod_cern_meta` | (E) | 允许Apache使用CERN httpd元文件，从而可以在发送文件时对头进行修改 | | `mod_charset_lite` | (X) | 允许对页面进行字符集转换 | | `mod_dav` | (E) | 允许Apache提供DAV协议支持 | | `mod_dav_fs` | (E) | 为 `mod_dav` 访问服务器上的文件系统提供支持 | | `mod_dav_lock` | (E) | 为 `mod_dav` 锁定服务器上的文件提供支持 | | `mod_dbd` | (E) | 管理SQL数据库连接，为需要数据库功能的模块提供支持 | | `mod_deflate` | (E) | 压缩发送给客户端的内容 | | `mod_disk_cache` | (E) | 基于磁盘的缓冲管理器 | | `mod_dumpio` | (E) | 将所有I/O操作转储到错误日志中 | | `mod_echo` | (X) | 一个很简单的协议演示模块 | | `mod_example` | (X) | 一个很简单的Apache模块API演示模块 | | `mod_expires` | (E) | 允许通过配置文件控制HTTP的" `Expires:` "和" `Cache-Control:` "头内容 | | `mod_ext_filter` | (E) | 使用外部程序作为过滤器 | | `mod_file_cache` | (X) | 提供文件描述符缓存支持，从而提高Apache性能 | | `mod_headers` | (E) | 允许通过配置文件控制任意的HTTP请求和应答头信息 | | `mod_ident` | (E) | 实现RFC1413规定的ident查找 | | `mod_info` | (E) | 生成Apache配置情况的Web页面 | | `mod_ldap` | (E) | 为其它LDAP模块提供LDAP连接池和结果缓冲服务 | | `mod_log_forensic` | (E) | 实现"对比日志"，即在请求被处理之前和处理完成之后进行两次记录 | | `mod_logio` | (E) | 对每个请求的输入/输出字节数以及HTTP头进行日志记录 | | `mod_mem_cache` | (E) | 基于内存的缓冲管理器 | | `mod_mime_magic` | (E) | 通过读取部分文件内容自动猜测文件的MIME类型 | | `mod_proxy` | (E) | 提供HTTP/1.1的代理/网关功能支持 | | `mod_proxy_ajp` | (E) | `mod_proxy` 的扩展，提供Apache JServ Protocol支持 | | `mod_proxy_balancer` | (E) | `mod_proxy` 的扩展，提供负载平衡支持 | | `mod_proxy_connect` | (E) | `mod_proxy` 的扩展，提供对处理HTTP `CONNECT` 方法的支持 | | `mod_proxy_ftp` | (E) | `mod_proxy` 的FTP支持模块 | | `mod_proxy_http` | (E) | `mod_proxy` 的HTTP支持模块 | | `mod_rewrite` | (E) | 一个基于一定规则的实时重写URL请求的引擎 | | `mod_so` | (E) | 允许运行时加载DSO模块 | | `mod_speling` | (E) | 自动纠正URL中的拼写错误 | | `mod_ssl` | (E) | 使用安全套接字层(SSL)和传输层安全(TLS)协议实现高强度加密传输 | | `mod_suexec` | (E) | 使用与调用web服务器的用户不同的用户身份来运行CGI和SSI程序 | | `mod_unique_id` | (E) | 为每个请求生成唯一的标识以便跟踪 | | `mod_usertrack` | (E) | 使用Session跟踪用户(会发送很多Cookie)，以记录用户的点击流 | | `mod_version` | (E) | 提供基于版本的配置段支持 | | `mod_vhost_alias` | (E) | 提供大批量虚拟主机的动态配置支持 |

多路处理模块(MPM)

必须有而且只能有一个MPM被静态包含进核心，你可以使用下面的配置选项进行选择：

`--with-mpm=MPM`

其中，`MPM` 是你想要使用的多路处理模块的名字。如果你不使用这个选项，那么将会使用对应于各平台的默认MPM，可选的MPM如下：`beos`，`mpmt_os2`，`prefork`，`worker`

第三方模块

有至少两种方法可以添加第三方模块，最简单的方法是作为配置参数提供，语法如下：

`--with-module=module-type:module-file[, module-type:module-file]`

`module-file` 是模块的源代码文件名，该文件必须位于Apahe源代码目录树的"`modules/module-type`"目录下，如果 `configure` 没有在那里找到module-file，则将它看作一个绝对路径名并尝试将其复制到"`modules/module-type`"目录中，如果"`modules/module-type`"目录不存在，`configure` 将新建一个"`modules/module-type`"目录并在其中放置一个标准的 `Makefile.in` 文件。这种方法有两个明显的缺陷：

1. 模块的源代码必须是单一文件
2. 模块只能静态连接进核心，而不能作为DSO模块

所以一般并不使用此方法，而是使用 `apxs` (Apache扩展工具)来添加第三方模块支持。

杂项选项

`--enable-nonportable-atomics`

若只打算在486以上的CPU上运行Apache，那么使用该选项可以启用更加高效的基于互斥执行的原子操作。

`--enable-v4-mapped`

使用相同的套接字同时处理IPv4和IPv6的连接，也就是启用地址映射。在FreeBSD、NetBSD、OpenBSD以外的平台上是默认值。

`--disable-v4-mapped`

使用不同的套接字分别处理IPv4和IPv6的连接，也就是禁用地址映射。在FreeBSD、NetBSD、OpenBSD上是默认值。

`--enable-maintainer-mode`

使用所有警告和调试符号编译源代码，请勿用于正式服务器，它会影响性能。

`--enable-exception-hook`

允许在子进程崩溃以后启用一个钩子来运行异常处理程序。参见 `EnableExceptionHook` 指令

`--with-port=PORT`

设定 `httpd` 的默认的监听端口[默认为：80]，该值仅在生成默认配置文件 `httpd.conf` 时使用。

`--with-program-name=NAME`

指定可执行程序的名字[默认为： `httpd`]，若使用此选项则默认配置文件的名字将同时变成"NAME.conf"。

传递给 `apr-config` 脚本的选项

译者注：下述三个选项并未出现在官方手册中，译者不保证其真实性，译者本人亦未使用过，仅供有兴趣的玩家参考。

`--disable-threads`

禁用线程支持，如果不使用线程化的MPM，可以关闭它以减少系统开销。

`--disable-ipv6`

禁用IPv6支持

`--disable-dso`

禁用DSO支持

特殊程序包选项

这些指令用于定义特殊程序包相关的选项。

`--with-apr=DIR|FILE`

Apache可移植运行时(APR)是httpd源码的一部分并会自动与httpd一起创建。如果你想使用一个已经存在的APR，就必须在这里指定 `apr-config` 脚本的路径。可以使用此脚本的绝对路径或已有的APR安装目录(`apr-config` 必须位于此目录或者其下的" `bin` "子目录中)。

`--with-apr-util=DIR|FILE`

Apache可移植运行时工具包(APU)是httpd源码的一部分并会自动与httpd一起创建。如果你想使用一个已经存在的APU，就必须在这里指定 `apu-config` 脚本的路径。可以使用此脚本的绝对路径或已有的APU安装目录(`apu-config` 必须位于此目录或者其下的" `bin` "子目录中)。

`--with-ssl=DIR`

如果启用了 `mod_ssl` ， `configure` 脚本将会自动搜寻已经安装的OpenSSL，你可以在这里指定SSL/TLS工具包的安装路径。

`--with-z=DIR`

如果你启用了压缩模块(比如 `mod_deflate`)， `configure` 脚本将会自动搜寻已经安装的 `zlib` 库，你可以在这里指定它的安装路径。

`--with-perl=DIR`

有些用Perl写的支持脚本，如 `apxs` 或 `dbmmanage` ，需要Perl5解释器(5.003或以上的版本就足够了)。如果系统中存在多个Perl解释器，比如有系统提供的Perl 4 ，还有你自己安装的Perl 5 ，推荐你使用该选项来指定正确的版本。如果没有Perl 5也没关系，这并不影响Apache httpd的编译和安装，只是相关的支持脚本不能使用而已。

`--with-pcre=DIR`

5.0版的Perl兼容正则表达式库(PCRE)已经被包含进来了，如果你想使用系统中已经安装好的PCRE ，就可以在这里指定其安装路径。

`--with-ldap=DIR`

一些Apache模块，比如 `mod_ldap` 和 `mod_authnz_ldap` 需要APU支持LDAP(默认并不支持)，只要使用其中之一，就要使用该选项指定LDAP的安装路径。

一些Apache模块，比如 `mod_authn_dbm` 和 `mod_rewrite` 需要使用DBM数据库，APU中已经包含了SDBM ，所以这个数据库总是可用的。如果你想使用其他类型的数据库，就要使用以下选项：

`--with-gdbm[=path]`

使用GNU DBM代替SDBM；如果不指定path ，则 `configure` 脚本将会在默认路径上搜索GNU DBM的包含文件和库的位置。如果指定path ，则 `configure` 脚本会在 `path/lib` 和 `path/include` 目录中搜索GNU DBM的包含文件和库。还可以使用"inc-path:lib-path"的形式分别指定GNU DBM的包含文件和库的位置。

`--with-ndbm[=path]`

使用New DBM代替SDBM；如果不指定path ，则 `configure` 脚本将会在默认路径上搜索New DBM的包含文件和库的位置。如果指定path ，则 `configure` 脚本会在 `path/lib` 和 `path/include` 目录中搜索New DBM的包含文件和库。还可以使用"inc-path:lib-path"的形式分别指定New DBM的包含文件和库的位置。

`--with-berkeley-db[=path]`

使用Berkeley DB代替SDBM；如果不指定path ，则 `configure` 脚本将会在默认路径上搜索Berkeley DB的包含文件和库的位置。如果指定path ，则 `configure` 脚本会在 `path/lib` 和 `path/include` 目录中搜索Berkeley DB的包含文件和库。还可以使用"inc-path:lib-path"的形式分别指定Berkeley DB的包含文件和库的位置。

注意

DBM数据库选项是由APU提供并传递给APU配置脚本的。所以如果使用 `--with-apr-util` 指定一个已安装的APU来代替，那么这些选项便无效。你可以同时使用几种不同的DBM实现，然后使用运行时配置动态选择其中之一。

支持程序选项

`--enable-static-support`

使用静态连接(默认为动态连接)编译所有二进制支持程序。若不使用该选项也可以使用下面的选项分别指定每个支持程序：

`--enable-static-ab`

使用静态连接编译 `ab`

`--enable-static-checkgid`

使用静态连接编译 `checkgid`

`--enable-static-htdbm`

使用静态连接编译 `htdbm`

`--enable-static-htdigest`

使用静态连接编译 `htdigest`

`--enable-static-htpasswd`

使用静态连接编译 `htpasswd`

`--enable-static-logresolve`

使用静态连接编译 `logresolve`

`--enable-static-rotatelogs`

使用静态连接编译 `rotatelogs`

suexec配置选项

`--enable-suexec`

使用这个选项以启用 `suexec` ，它可以允许你为CGI程序指定uid和gid 。如果你不精通**suexec**的工作机制，请不要使用它！

仅在启用了上述选项的情况下，才可以使用以下选项微调 `suexec` 的各种特性。方括号"[]"内是默认值。参见配置和安装suEXEC以获得更多信息。

`--with-suexec-bin`

`suexec` 二进制文件目录[`--sbindir`]

`--with-suexec-caller`

允许调用 `suexec` 的用户，必须和运行 `httpd` 子进程的用户相同。

`--with-suexec-docroot`

允许 `suexec` 对其中的文件具有执行权限的根目录[`--datadir/htdocs`]

`--with-suexec-gidmin`

允许执行 `suexec` 的最小GID[100]

`--with-suexec-logfile`

`suexec` 日志文件名[默认文件名为： `suexec_log` ， 位于 `--logfiledir` 目录下]

`--with-suexec-safepath`

对 `suexec` "安全"的 `PATH` 环境变量的值[`/usr/local/bin:/usr/bin:/bin`]

`--with-suexec-userdir`

用户主目录下允许 `suexec` 对其中的文件具有执行权限的子目录，仅在将 `suexec` 和用户网站目录(由 `mod_userdir` 提供支持)一起使用的情况下才需要设置此选项。[`public_html`]

`--with-suexec-uidmin`

允许执行 `suexec` 的最小UID[100]

`--with-suexec-umask`

`suexec` 进程的 `umask` [取决于系统的设定]

环境变量

可以通过指定某些环境变量来修改 `configure` 脚本的默认选择，或者帮助 `configure` 脚本找到名字和/或位置不标准的库和程序。

`CC`

C编译器

`CFLAGS`

C编译器的flags

`CPP`

C预处理程序

`CPPFLAGS`

C/C++预处理程序flags，比如使用" `-Iincludedir` "指定一个非标准的头文件目录includedir 。

`LDFLAGS`

连接器flags，比如使用"-L `-Llibdir` "指定一个非标准的库文件目录libdir 。

dbmmanage - 管理DBM格式的用户认证文件

`dbmmanage` 建立和更新存储用户名和密码的DBM格式的文件，以用于 `mod_authn_dbm` 对HTTP用户进行基本认证。Apache HTTP服务器上的有效资源可以被限制为仅允许由 `dbmmanage` 建立的文件中的用户所访问。此程序仅用于用户名是存储在一个DBM文件中的情况下，如果使用文本数据库，请参见 `htpasswd` 。

本手册页仅列出命令行参数，配置用户认证的相关信息请参见 `认证、授权、访问控制` 文档。

语法

`**dbmmanage** [ encoding ] filename add|adduser|check|delete|update username [ encpasswd [`

`**dbmmanage** filename view [ username ]`

`**dbmmanage** filename import`

选项

`filename`

DBM格式文件的文件名。一般不带 `.db` , `.pag` , `.dir` 后缀。

`username`

操作所针对的用户。username中不能有冒号(`:`)。

`encpasswd`

这是已经加密的密码，用于 `update` 和 `add` 命令。使用一个连字符(`-`)可以显示输入密码的提示，然后输入。另外，在用于 `update` 命令时，使用一个句号(`.`)可以保持原有密码不变。

`group`

用户所属的组名，组名中不能有冒号(`:`)。如果不希望指定该用户所属的组，可以使用一个连字符(`-`)，但是需要填写comment项。另外，在用于 `update` 命令时，使用一个句号(`.`)可以保持原来所属的组不变。

`comment`

这是对该用户的说明，如真实姓名、邮件地址之类。服务器本身并不使用此信息。

编码

`-d`

crypt 加密(在Windows和Netware以外平台上的默认值)

`-m`

MD5 加密(在Windows和Netware平台上的默认值)

`-s`

SHA1 加密

`-p`

纯文本(不推荐)

命令

`add`

在filename中增加一个包含了username和已加密密码encpasswd的项。

```
dbmmanage passwords.dat add rbowen foKntnEF3KSXA
```

`adduser`

要求输入密码，然后在filename中增加一个username项。

```
dbmmanage passwords.dat adduser krietz
```

`check`

要求输入密码，然后检查filename中是否存在username并且其密码与输入的相同。

```
dbmmanage passwords.dat check rbowen
```

`delete`

在filename中删除username项。

```
dbmmanage passwords.dat delete rbowen
```

`import`

从 `STDIN` 读取 `username:password` 的信息(每行一对)，然后增加到filename中。其中的密码必须是已加密的。

`update`

类似 `adduser` 命令，但是它可以确认username已经存在于filename中。

```
dbmmanage passwords.dat update rbowen
```

`view`

仅显示DBM文件的内容。如果指定了username则仅显示该用户的信息。

```
dbmmanage passwords.dat view
```

Bugs

注意，实际上存在有许多不同的DBM文件格式，你的系统中也可能存在不止一种的支持库，常见的有SDBM, NDBM, GDBM, Berkeley DB 2 。麻烦的是，所有这些库都使用了不同的文件格式，因而你必须确保filename所采用的格式能够为 `dbmmanage` 所接受。目前， `dbmmanage` 无法自己确定所查找的文件的DBM类型。如果使用了错误的格式，则简单返回nothing ，或者建立一个不同名称的不同的DBM文件，而最坏的情况是，在试图写入这个文件时，可能会破坏该DBM文件。

`dbmmanage` 有一个DBM格式参数选择列表，在程序前部由 `@AnyDBM::ISA` 数组定义。由于我们更喜欢 Berkeley DB 2 格式， `dbmmanage` 查找系统库的顺序是：Berkeley DB 2, NDBM, GDBM, SDBM 。 `dbmmanage` 会使用第一个找到的库来处理所有的DBM文件操作。此顺序与perl中标准的 `@AnyDBM::ISA` 的顺序略微不同，所以，如果要使用任何其他工具来管理DBM文件，则必须确保该工具是按此顺序处理的。在用其他语言比如C的程序来处理这些文件时，也要考虑这一点。

在大多数Unix系统中，都可以用 `file` 程序来查看DBM文件的格式。

htcacheclean - 清理磁盘缓冲区

`htcacheclean` 可以用于将 `mod_disk_cache` 的磁盘缓冲区占用的空间保持在一个合理的水平。这个工具可以手动运行也可以作为后台守护进程运行。当作为守护进程运行的时候，它将每隔一段时间检查一次缓冲区所在目录并进行清理。你可以通过 TERM 或 INT 信号停止守护进程的清理操作。

语法

```
**htcacheclean** [ -**D** ] [ -**v** ] [ -**t** ] [ -**r** ] [ -**n** ] -**p**path -**l**l
```

```
**htcacheclean** -**b** [ -**n** ] [ -**t** ] [ -**i** ] -**d**interval -**p**path -**l**l
```

选项

`-dinterval`

每隔interval分钟进行一次清理。这个选项和 `-D` , `-v` , `-r` 互斥，不能同时使用。要关闭清理进程，可以使用 `SIGTERM` 或 `SIGINT` 信号。

`-D`

进行一次"演习"而不真正清理任何内容。这个选项和 `-d` 互斥，不能同时使用。

`-v`

显示详细的统计信息。这个选项和 `-d` 互斥，不能同时使用。

`-r`

进行彻底的清理。它假定Apache web服务器已经停止(否则你将在缓冲区中留下垃圾)。这个选项和 `-d` 互斥，不能同时使用。同时该选项隐含了 `-t` 选项。

`-n`

温和精细的清理。这样清理过程将会被减慢以有利于其它进程的执行。 `htcacheclean` 有时将会进入休眠状态，以便：(a)磁盘IO被延时； (b)操作系统内核可以同时执行其它进程。

`-t`

删除所有空目录，而默认只删除缓存文件。因为在某些配置情况下会建立数量巨大的目录，这样很可能导致inode或文件分配表耗尽。我们建议你使用这个选现。

`-ppath`

将path指定为磁盘缓冲区的根目录。它必须和 `CacheRoot` 指定的目录相同。

`-llimit`

将limit指定为磁盘缓冲区允许占用的最大空间。用 `xxB` 表示xx字节，用 `xxK` 表示xx千字节，用 `xxM` 表示xx兆字节。

`-i`

智能运行，也就是仅在磁盘缓冲区的内容被更改的情况下运行。仅能够和 `-d` 选项同时使用。

退出状态

`htcacheclean` 仅在一切操作都成功的情况下返回" `0` "，否则返回" `1` "。

htdbm - 操作DBM密码数据库

`htdbm` 用于操作由 `mod_authn_dbm` 提供的HTTP基本认证所使用的保存用户名和密码的DBM数据库文件。参见 `dbmmanage` 文档以获得这些DBM文件的更多信息。

语法

```
**htdbm** [ -**T**DBTYPE ] [ -**c** ] [ -**m** | -**d** | -**p** | -**s** ] [ -**t** ] [ -
```

```
**htdbm** -**b** [ -**T**DBTYPE ] [ -**c** ] [ -**m** | -**d** | -**p** | -**s** ] [ -**t*
```

```
**htdbm** -**n** [ -**c** ] [ -**m** | -**d** | -**p** | -**s** ] [ -**t** ] [ -**v** ] use
```

```
**htdbm** -**nb** [ -**c** ] [ -**m** | -**d** | -**p** | -**s** ] [ -**t** ] [ -**v** ] u
```

```
**htdbm** -**v** [ -**T**DBTYPE ] [ -**c** ] [ -**m** | -**d** | -**p** | -**s** ] [ -**t*
```

```
**htdbm** -**vb** [ -**T**DBTYPE ] [ -**c** ] [ -**m** | -**d** | -**p** | -**s** ] [ -**t
```

```
**htdbm** -**x** [ -**T**DBTYPE ] [ -**m** | -**d** | -**p** | -**s** ] filename username
```

```
**htdbm** -**l** [ -**T**DBTYPE ]
```

选项

`-b`

使用批处理方式。也就是直接从命令行获取密码而不进行提醒。使用这个选项需要特别注意，因为命令行中的密码是清晰可见的。

`-c`

创建passwdfile文件。如果passwdfile已经存在，那么将被清空并改写。该选项不能和 `-n` 同时使用。

`-n`

在标准输出上显示结果而不是更新数据库。这个选项改变了命令行语法，因为passwdfile参数(通常是第一个)被忽略了。该选项不能和 `-c` 同时使用。

`-m`

使用MD5加密密码。在Windows, Netware, TPF上这是默认方法。

`-d`

使用 `crypt()` 对密码进行加密。在Windows, Netware, TPF以外的平台上这是默认方法。虽然有可能在所有的平台上被 `htdbm` 支持，但是在Windows, Netware, TPF上，该方法不能被 `httpd` 所支持。

`-s`

使用 SHA 对密码进行加密。这种方法易于通过LDAP目录交换格式和Netscape server进行交换。

`-p`

使用明文密码(不加密)。虽然 `htdbm` 在所有平台上都支持这种方法，但是 `httpd` 只能在Windows, Netware, TPF上支持这种方法。

`-l`

在标准输出上显示每一个用户名以及对应的注释。

`-t`

将最后一个参数解释为注释。指定这个选项可以在命令行中添加一个额外的字符串，这个字符串可以被存储在数据库中对应用户名的"Comment"字段中。

`-v`

校验用户名和密码。程序将会显示一个密码是否正确的信息。如果密码不正确程序退出时的错误代码将是"3"。

`-x`

删除用户。如果指定的用户名存在与数据库中，则删除该用户。

`filename`

DBM文件的文件名。通常不包含 `.db` ，`.pag` ，`.dir` 后缀。如果同时使用了 `-c` 选项，若DBM文件已存在则更新它，若不存在则创建它。

`username`

在passwdfile中添加或更新记录。若username不存在则添加一条记录，若存在则更新其密码。

`password`

将要被加密存储的明文密码。仅与 `-b` 一同使用。

`-TDBTYPE`

DBM文件的类型(SDBM, GDBM, DB, "default")。

Bugs

实际上存在有许多不同的DBM文件格式，你的系统中也可能存在不止一种的支持库，常见的有SDBM, NDBM, GNU GDBM, Berkeley/Sleepycat DB 2/3/4 。麻烦的是，所有这些库都使用了不同的文件格式，因而你必须确保filename所采用的格式能够为 `htdbm` 所接受。目前， `htdbm` 无法自己确定所查找的文件的DBM类型。如果使用了错误的格式，则简单返回nothing ，或者建立一个不同名称的不同的DBM文件，而最坏的情况是，在试图写入这个文件时，可能会破坏该DBM文件。

在大多数Unix系统中，都可以用 `file` 程序来查看DBM文件的格式。

返回值

`htdbm` 仅在用户名和密码被成功存入数据库或成功更新的情况下返回" `0` "。若访问文件发生错误则返回" `1` "；若命令行语法错误则返回" `2` "；若密码验证失败则返回" `3` "；若正在进行中的操作被打断则返回" `4` "；若值(username, filename, password, 计算结果)长度超标则返回" `5` "；若用户名包含非法字符(参见限制)则返回" `6` "；若指定的文件不能被正确识别则返回" `7` "。

示例

```
htdbm /usr/local/etc/apache/.htdbm-users jsmith
```

添加或修改用户 `jsmith` 的密码。密码将被提示输入。在Windows平台上，密码将使用Apache修改过的MD5算法进行加密；在其它平台上将使用 `crypt()` 进行加密。如果指定的文件不存在， `htdbm` 将只返回一个错误代码，而不做其它任何事。

```
htdbm -c /home/doe/public_html/.htdbm jane
```

创建一个新文件并在其中添加一条用户 `jane` 的记录。密码将被提示输入。如果文件存在但是不能被读取或写入，则不会有任何记录被修改，同时 `htdbm` 将会显示一个错误信息并返回一个错误代码。

```
htdbm -mb /usr/web/.htdbm-all jones Pwd4Steve
```

将来自命令行的密码(`Pwd4Steve`)使用MD5算法加密，并将其存入指定的文件。

安全方面的考虑

Web密码文件(比如由 `htdbm` 管理)不应当存在于网络空间中，即不能被客户端有机会访问。

我们反对使用 `-b` 选项，因为密码将以明文的形式出现在命令行中。

限制

在Windows和MPE平台上，用 `htdbm` 加密的密码最大长度是 `255` 字符。超出部分将被截断。

`htdbm` 使用的MD5加密算法已经被Apache修改过了，仅能够被Apache识别，不能被其它Web服务器识别。

用户最大长度是 `255` 字节，并且不能包含冒号(`:`)。

htdigest - 管理用于摘要认证的用户文件

`htdigest` 建立和更新用于摘要认证的、存储用户名/域/密码的文本文件。服务器上的资源可以被限制为仅允许由 `htdigest` 建立的文件中的用户访问。

本手册页仅列出命令行参数，配置摘要认证的相关指令的细节请参见 `mod_auth_digest` 文档。

语法

`**htdigest** [ -**c** ] passwdfile realm username`

选项

`-c`

建立passwdfile 。如果passwdfile已经存在，则会首先把它删除。

`passwdfile`

包含用户名/域/密码的文件名。若指定了 `-c` ，那么，如果该文件不存在则新建，否则先删除然后再新建。

`realm`

该用户名所属的域。

`username`

在passwdfile中所建立或更新的用户名。如果username不存在，则增加一项，否则改变其密码。

httxt2dbm - 生成RewriteMap指令使用的dbm文件

`httxt2dbm` 可以用来从输入的文本生成RewriteMap指令使用的dbm文件(使用 `dbm` 映射类型)。

语法

`**httxt2dbm** [ -**v** ] [ -**f** DBM_TYPE ] -**i** SOURCE_TXT -**o** OUTPUT_DBM`

选项

`-v`

输出更详细的信息。

`-f`

指定输出使用的DBM类型。若没有指定，将使用APR的默认类型。可用的类型有： `GDBM` GDBM 文件 `SDBM` SDBM 文件 `DB` berkeley DB 文件 `NDBM` NDBM 文件 `default` 使用默认类型

`-i`

用于创建dbm文件的输入文件。该输入文件的格式必须是每行一条记录，其格式如下： `key value` 参考 `RewriteMap` 文档以获得关于这个文件格式更多说明。

`-o`

输出的dbm文件名。

例子

```
httxt2dbm -i rewritemap.txt -o rewritemap.dbm

httxt2dbm -f SDBM -i rewritemap.txt -o rewritemap.dbm
```

htpasswd - 管理用于基本认证的用户文件

`htpasswd` 建立和更新用于基本认证的存储用户名/密码的文本文件。如果 `htpasswd` 不能读写此文件，它返回一个出错代码，而不做任何修改。

服务器上的资源可以被限制为仅允许由 `htpasswd` 建立的文件中的用户所访问。此程序只能管理存储在文本文件中的用户名和密码，但是它可以加密并显示密码信息，从而可以为其他数据存储类型所利用。要使用DBM数据库，请参见 `dbmmanage` 。

`htpasswd` 使用专为Apache作了修改的MD5算法或系统函数 `crypt()` 加密密码。 `htpasswd` 所管理的文件可以包含两种类型的密码；有些用户的密码使用MD5加密的，而同一个文件中的其他用户密码则使用 `crypt()` 加密。

本手册页仅列出命令行参数，配置基本认证的相关指令的细节请参见 `mod_auth_basic` 文档。

语法

`**htpasswd** [ -**c** ] [ -**m** ] [ -**D** ] passwdfile username`

`**htpasswd** -**b** [ -**c** ] [ -**m** | -**d** | -**p** | -**s** ] [ -**D** ] passwdfile`

`**htpasswd** -**n** [ -**m** | -**d** | -**s** | -**p** ] username`

`**htpasswd** -**nb** [ -**m** | -**d** | -**s** | -**p** ] username password`

选项

`-b`

使用批处理方式。也就是直接从命令行获取密码而不进行提醒。使用这个选项需要特别注意，因为命令行中的密码是清晰可见的。

`-c`

创建passwdfile文件。如果passwdfile已经存在，那么将被清空并改写。该选项不能和 `-n` 同时使用。

`-n`

在标准输出上显示结果而不是更新文件。用于生成可以为Apache非文本输出存储格式所接受的密码记录。这个选项改变了命令行语法，因为passwdfile参数(通常是第一个)被忽略了。该选项不能和 `-c` 同时使用。

`-m`

使用MD5加密密码。在Windows, Netware, TPF上这是默认方法。

`-d`

使用 `crypt()` 对密码进行加密。在Windows, Netware, TPF以外的平台上这是默认方法。虽然有可能在所有的平台上被 `htpasswd` 支持，但是在Windows, Netware, TPF上，该方法不能被 `httpd` 所支持。

`-s`

使用 SHA 对密码进行加密。这种方法易于通过LDAP目录交换格式和Netscape server进行交换。

`-p`

使用明文密码(不加密)。虽然 `htpasswd` 在所有平台上都支持这种方法，但是 `httpd` 只能在Windows, Netware, TPF上支持这种方法。

`-D`

如果 `username` 存在于 `passwdfile` 中，则删除该用户。

`passwdfile`

包含用户名和密码的文本文件的名称。如果使用了 `-c` 选项，若文件已存在则更新它，若不存在则创建它。

`username`

在passwdfile中添加或更新记录。若username不存在则添加一条记录，若存在则更新其密码。

`password`

将被加密并存储到文件中的明文密码。必须和 `-b` 同时使用。

返回值

`htpasswd` 仅在用户名和密码被成功存入passwdfile或成功更新的情况下返回" `0` "。若访问文件发生错误则返回" `1` "；若命令行语法错误则返回" `2` "；若密码验证失败则返回" `3` "；若正在进行中的操作被打断则返回" `4` "；若值(username, filename, password, 计算结果)长度超标则返回" `5` "；若用户名包含非法字符(参见限制)则返回" `6` "；若指定的文件不能被正确识别则返回" `7` "。

示例

```
htpasswd /usr/local/etc/apache/.htpasswd-users jsmith
```

添加或修改用户 `jsmith` 的密码。密码将被提示输入。在Windows平台上，密码将使用Apache修改过的MD5算法进行加密；在其它平台上将使用 `crypt()` 进行加密。如果指定的文件不存在， `htpasswd` 将只返回一个错误代码，而不做其它任何事。

```
htpasswd -c /home/doe/public_html/.htpasswd jane
```

创建一个新文件并在其中添加一条用户 `jane` 的记录。密码将被提示输入。如果文件存在但是不能被读取或写入，则不会有任何记录被修改，同时 `htpasswd` 将会显示一个错误信息并返回一个错误代码。

```
htpasswd -mb /usr/web/.htpasswd-all jones Pwd4Steve
```

将来自命令行的密码(`Pwd4Steve`)使用MD5算法加密，并将其存入指定的文件。

安全方面的考虑

Web密码文件(比如由 `htpasswd` 管理)不应当存在于网络空间中，即不能被客户端有机会访问。

我们反对使用 `-b` 选项，因为密码将以明文的形式出现在命令行中。

限制

在Windows和MPE平台上，用 `htdbm` 加密的密码最大长度是 `255` 字符。超出部分将被截断。

`htdbm` 使用的MD5加密算法已经被Apache修改过了，仅能够被Apache识别，不能被其它Web服务器识别。

用户最大长度是 `255` 字节，并且不能包含冒号(`:`)。

logresolve - 解析Apache日志中的IP地址为主机名

`logresolve` 是一个解析Apache访问日志中IP地址的后处理程序。为了使对名称服务器的影响降到最低，它拥有极为自主的内部散列表缓存，使每个IP值仅仅在第一次从日志文件中读出时才被解析一次。

此程序从标准输入设备上获得需要解析的Apache日志文件，其中的IP地址必须在每行的开始处，行中其余信息必须以空格分隔。

语法

```
**logresolve** [ -**s** filename ] [ -**c** ] &lt; access_log &gt; access_log.new
```

选项

`-s filename`

指定记录统计信息的文件名。

`-c`

此选项使 `logresolve` 执行DNS验证：在把IP地址解析为主机名后，按主机名查找IP地址，以验证原地址能否与其中之一相匹配。

rotatelogs - 滚动Apache日志的管道日志程序

`rotatelogs` 是一个配合Apache管道日志功能使用的简单程序。举例：

```
CustomLog "|bin/rotatelogs /var/logs/logfile 86400" common
```

此配置会建立文件"/var/logs/logfile.nnnn"，其中的nnnn是名义上的日志启动时的系统时间(此时间总是滚动时间的倍数，可以用于cron脚本的同步)。在滚动时间到达时(在此例中是24小时以后)，会产生一个新的日志。

```
CustomLog "|bin/rotatelogs /var/logs/logfile 5M" common
```

此配置会在日志文件大小增长到5兆字节时滚动该日志。

```
ErrorLog "|bin/rotatelogs /var/logs/errorlog.%Y-%m-%d-%H_%M_%S 5M"
```

此配置会在错误日志大小增长到5兆字节时滚动该日志，日志文件名后缀会按照如下格式创建：`errorlog.YYYY-mm-dd-HH_MM_SS` 。

语法

`**rotatelogs** [ -**l** ] logfile [ rotationtime [ offset ]] | [ filesizeM ]`

选项

`-l`

使用本地时间代替GMT时间作为时间基准。注意：在一个改变GMT偏移量(比如夏令时)的环境中使用 `-l` 会导致不可预料的结果。

`logfile`

它加上基准名就是日志文件名。如果logfile中包含"%"，则它会被视为用于 `strftime()` 的格式字符串；否则它会被自动加上以秒为单位的".nnnnnnnnnn"后缀。这两种格式都表示新的日志开始使用的时间。

`rotationtime`

日志文件滚动的以秒为单位的间隔时间。

`offset`

相对于UTC的时差的分钟数。如果省略，则假定为"0"并使用UTC时间。比如，要指定UTC时差为"-5小时"的地区的当地时间，则此参数应为" `-300` "。

`filesizeM`

指定以 `filesizeM` 文件大小滚动，而不是按照时间或时差滚动。

可移植性

下列日志文件格式字符串可以为所有的 `strftime()` 实现所支持，见各种扩展库对应的 `strftime()` 的手册。

| `%A` | 星期名全称(本地的) | | `%a` | 3个字符的星期名(本地的) | | `%B` | 月份名的全称(本地的) | | `%b` | 3个字符的月份名(本地的) | | `%c` | 日期和时间(本地的) | | `%d` | 2位数的一个月中的日期数 | | `%H` | 2位数的小时数(24小时制) | | `%I` | 2位数的小时数(12小时制) | | `%j` | 3位数的一年中的日期数 | | `%M` | 2位数的分钟数 | | `%m` | 2位数的月份数 | | `%p` | am/pm12小时制的上下午(本地的) | | `%S` | 2位数的秒数 | | `%U` | 2位数的一年中的星期数(星期天为一周的第一天) | | `%W` | 2位数的一年中的星期数(星期一为一周的第一天) | | `%w` | 1位数的星期几(星期天为一周的第一天) | | `%X` | 时间(本地的) | | `%x` | 日期(本地的) | | `%Y` | 4位数的年份 | | `%y` | 2位数的年份 | | `%Z` | 时区名 | | `%%` | 符号"%"本身 |

suexec - 在执行外部程序之前切换用户

`suexec` 使Apache HTTP服务器在执行CGI程序之前切换为另一个用户。为此，它必须以 `root` 身份运行。由于Apache监听子进程一般不是以 `root` 身份运行的，所以，可执行文件 `suexec` 的所有者必须是 `root` ，而且其setuid位必须被设置。除了 `root` ，这个文件绝不应该对其他人开放写权限。

有关涉及的概念和suexec安全模型方面的信息，参见suexec文档)。

语法

`**suexec** -**V**`

选项

`-V`

如果你是 `root` ，此选项可以显示 `suexec` 的编译选项。由于安全方面的原因，所有的配置选项只能在编译时确定。

其他程序

以下是包含在Apache中的没有专门手册页的简单支持程序，这些程序不是自动安装的，而是在配置过程以后，装在"`support`"目录下的。

log_server_status

此perl脚本可以由频繁使用的诸如cron的工具所调用。它连接到服务器并下载状态信息，并格式化此信息为一行，并记录在一个文件中。要指定结果的输出文件的位置，可以调整该脚本首行中的变量。

split-logfile

此perl脚本可以把记录了多个web服务器的存取日志文件内容切分为独立的文件。它假设其中每行的第一个字段是虚拟主机标识符(使用"`%v`")，并在当前目录中生成命名为虚拟主机标识符+"`.log`"的文件。

记录了多个web服务器的日志文件是从stdin中读取的。读出的记录会被追加在已经存在的日志文件中。

杂项文档

以下列出Apache web服务器开发项目的其他文档。

警告

这些文档没有完全包含Apache HTTP服务器2.2版本中已经更新的内容，有些内容可能仍然有效，使用中请注意。

性能方面的提示 — Apache优化

在运行时和编译时配置Apache以获得最佳性能的提示，及其原因。

安全方面的提示

应该做什么和不要做什么以保护Apache web站点安全。

URL重写指南

这是 `mod_rewrite` 的参考手册，描述了如何用 `mod_rewrite` 解决网站管理员在实践中常见的基于URL的典型问题。

相关标准

列出了大多数和Apache相关的标准。

与Apache相关的标准

本页列出了所有Apache遵守的相关标准，并伴有简要描述。

除了下面列出的参考信息，下列资源也值得看看：

- http://purl.org/NET/http-errata - HTTP/1.1 勘误表
- http://www.rfc-editor.org/errata.html - RFC 勘误表
- http://ftp.ics.uci.edu/pub/ietf/http/#RFC - 一个与HTTP相关的RFC列表

注意

本文档尚未全部完成

HTTP 推荐标准

不管使用了什么模块，Apache作为一个基本的web服务器都遵守以下IETF推荐标准：

RFC 1945 (Informational)

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. This documents HTTP/1.0.

RFC 2616 (Standards Track)

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This documents HTTP/1.1.

RFC 2396 (Standards Track)

A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource.

HTML 推荐标准

对于超文本标记语言(HTML)，Apache遵守以下IETF和W3C推荐标准：

RFC 2854 (Informational)

This document summarizes the history of HTML development, and defines the "text/html" MIME type by pointing to the relevant W3C recommendations.

HTML 4.01 Specification (Errata)

This specification defines the HyperText Markup Language (HTML), the publishing language of the World Wide Web. This specification defines HTML 4.01, which is a subversion of HTML 4.

HTML 3.2 Reference Specification

The HyperText Markup Language (HTML) is a simple markup language used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents.

XHTML 1.1 - Module-based XHTML (Errata)

This Recommendation defines a new XHTML document type that is based upon the module framework and modules defined in Modularization of XHTML.

XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) (Errata)

This specification defines the Second Edition of XHTML 1.0, a reformulation of HTML 4 as an XML 1.0 application, and three DTDs corresponding to the ones defined by HTML 4.

认证

对于不同的认证方法，Apache遵守以下IETF推荐标准：

RFC 2617 (Draft standard)

"HTTP/1.0", includes the specification for a Basic Access Authentication scheme.

语言/国家代码

以下链接记录了ISO和其他语言/国家代码信息：

ISO 639-2

ISO 639 provides two sets of language codes, one as a two-letter code set (639-1) and another as a three-letter code set (this part of ISO 639) for the representation of names of languages.

ISO 3166-1

These pages document the country names (official short names in English) in alphabetical order as given in ISO 3166-1 and the corresponding ISO 3166-1-alpha-2 code elements.

BCP 47 (Best Current Practice), RFC 3066

This document describes a language tag for use in cases where it is desired to indicate the language used in an information object, how to register values for use in this language tag, and a construct for matching such language tags.

RFC 3282 (Standards Track)

This document defines a "Content-language:" header, for use in cases where one desires to indicate the language of something that has RFC 822-like headers, like MIME body parts or Web documents, and an "Accept-Language:" header for use in cases where one wishes to indicate one's preferences with regard to language.

Apache模块

描述模块的术语

本文对用于描述模块的术语加以说明。

说明

对模块用途的简要说明。

状态(Status)

状态(Status)代表了此模块与Apache服务器结合的紧密程度；也就是说，有可能需要重新编译服务器以获得一个模块的功能。其可能的值有：

MPM

一个多路处理模块。与其他类型的模块不同的是，必须在编译配置时进行选择，必须有且仅有一个MPM被静态编译到服务器中。此类型的模块负责基本的对请求的处理和指派。

Base

默认被编译进服务器的模块，因此始终是有效的，除非你刻意在编译时从配置中删除此模块。

Extension

默认没有被编译进服务器的模块。要激活此模块并使用其功能，有可能需要修改服务器的编译配置并重新编译Apache 。

Experimental

实验性的模块，虽然作为Apache包的一部分提供，但是需要你自己承担使用中的风险。对此模块提供文档仅仅是为了保持完整性，而并不一定有技术支持。

External

不包含在Apache基本发行版中模块("第三方模块")。我们对此模块免责，并且不提供技术支持。

源代码文件

列出包含实现此模块的源代码文件，此名称也用于 `<IfModule>` 指令。

模块标识符

这个是 `LoadModule` 指令在动态加载模块时用于指定模块的标识符。其实，它就是源代码文件中拼写模块的外部变量名。

兼容性

如果该模块不是原始Apache2.0的发行版的一部分，此处会写明此模块应该被使用于哪个版本；另外，如果此模块在特定平台上有功能限制，此处也会有详细说明。

描述指令的术语

本文对用于描述Apache配置指令的术语加以说明。

说明

对指令用途的简单说明。

语法

说明该指令在配置文件中使用的形式(随指令的不同而不同)，在指令的定义中有说明。指令后面一般可以跟一个或多个用空格分开的参数。如果参数中有空格，则必须用双引号括起来，用方括号括起来的是可选参数。如果一个参数可以取多个值，则各个可能的值用"|"分开。应该原样输入的文字使用默认字体，而可变的必须按实际情况加以替换的会加强显示。使用可变参数个数的指令以"..."结尾，以表示最后一个参数可以重复。

指令的参数类型非常多，以下列出常用的部分。

URL

一个完整的包括类型、主机名和可选路径名的统一资源引用名，如： `http://www.example.com/path/to/file.html`

URL-path

*URL*中类型和主机名之后的部分，如" `/path/to/file.html` "是表示资源在网络空间(而不是文件系统)中的位置。

file-path

文件在本地文件系统中相对于根目录的路径，如" `/usr/local/apache/htdocs/path/to/file.html` "。除非以斜杠(/)开头，否则将被视为相对于ServerRoot的相对路径。

directory-path

目录在本地文件系统中相对于根目录的路径，如： `/usr/local/apache/htdocs/path/to/`

filename

不带路径信息的文件名，如： `file.html`

regex

Perl兼容的正则表达式，是对文本匹配模式的描述。指令的定义中会说明应该使用什么*regex*。

extension

一般是指*filename*中最后一个"."号后面的部分。不过，Apache可以辨认文件的多个*extension*，如果*filename*中含有多个"."，则第一个"."后面由每个"."分隔开的部分都是此文件的*extension*。比如"`file.html.en`"有两个*extension*：`.html`和`.en`。在Apache指令中指定*extension*时，可以有也可以没有前导的"."，而且不区分大小写。

MIME-type

一种用一个主格式类型和一个副格式类型并用斜杠分隔的描述文件格式的方法，如：`text/html`

env-variable

这是Apache配置过程中定义的环境变量的名称。注意，它不一定与操作系统中的环境变量相同。详情参见环境变量文档。

默认值

如果该指令有默认值(即如果你没有在配置中明确指定，那么Apache服务器会默认设置一个特定的值，并认为它是你设置的)，会在此处说明。如果没有，则会指明是"*None*"。注意，此处的默认值并不一定与服务器发行版中默认的httpd.conf中该指令的取值相同。

作用域

它表示该指令出现在配置文件的什么位置才是合法的。它是一个用逗号分隔的一个或多个下列值的列表：

server config

说明该指令可以用于服务器配置文件(`httpd.conf`)，但不能用于任何`<VirtualHost>`或`<Directory>`段以及`.htaccess`文件中。

virtual host

说明该指令可以用于服务器配置文件的`<VirtualHost>`段中。

directory

说明该指令可以用于服务器配置文件`<Directory>`，`<Location>`，`<Files>`，`<Proxy>`段中，并服从配置段一文的限制。

.htaccess

说明该指令可以用于针对单个目录及其子目录的 `.htaccess` 文件中。它可能会因overrides的设置而不起作用。

指令应该仅仅出现在允许出现的作用域中，否则会产生配置错误，并导致服务器不能正确处理请求，或者根本不能启动。

指令的有效位置，事实上是其所有被列出的作用域逻辑或的结果。也就是如果一个指令被标为" `server config, .htaccess` "则可以用于 `httpd.conf` 和 `.htaccess` ，但不能用于任何 `<Directory>` 或 `<VirtualHost>` 容器。

覆盖项

该属性表示要使 `.htaccess` 文件中的该指令有效必须激活的配置覆盖项。如果一个指令的作用域不包含 `.htaccess` ，则无此内容。

`AllowOverride` 指令使覆盖生效，并作用于一个特定的范围(比如一个目录)及其下分支，除非又被其下层中其他的 `AllowOverride` 指令所修改。对指令的说明中同时列出了其可能的覆盖项。

状态

状态代表了此指令与Apache服务器结合的紧密程度；也就是说，有可能需要重新编译服务器以获得一个指令的功能。其可能的值有：

Core

Apache服务器最核心的部分，始终有效。

MPM

由一个多路处理模块提供，此类指令仅仅在使用了指令定义中模块一行所列的MPM之一时才有效。

Base

由默认编译进服务器的一个Apache标准模块提供，一般总是有效的，除非你刻意在编译时从配置中删除此模块。

Extension

由一个默认不被编译进服务器的模块提供。要激活此指令并使用其功能，需要修改服务器编译时配置并重新编译Apache。

Experimental

由一个一般来说默认不被编译进服务器的模块提供，并且需要你自己承担使用中的风险。对此指令提供文档是为了保持完整性，而并不一定有技术支持。提供此指令的模块，是否默认被编译进入服务器都有可能，其说明页面的顶部注明了其有效性。

模块

对该指令提供支持的模块列表。

兼容性

如果该指令不是原始Apache2的发行版的一部分，此处会写明此指令应该被使用于哪个版本；另外，如果此指令在特定平台上有功能限制，此处会有详细说明。

Apache核心(Core)特性

说明	Apache HTTP服务器核心提供的功能，始终有效
状态	核心(C)

AcceptFilter 指令

说明	根据协议类型对监听Socket进行优化
语法	`AcceptFilter protocol accept_filter`
作用域	server config
状态	核心(C)
模块	core
兼容性	仅在 Apache 2.1.5 以后的版本中可用

这个指令使得操作系统根据协议类型对监听socket进行特别的优化。其基本前提是内核在数据接受完毕或一个完整的HTTP请求缓冲完成前不向服务器进程发送socket 。目前仅支持FreeBSD的接收过滤器(Accept Filter)和Linux的更原始的(more primitive) `TCP_DEFER_ACCEPT` 。

FreeBSD上的默认值是：

```
AcceptFilter http httpready
AcceptFilter https dataready
```

`httpready` 接收过滤器(Accept Filter)在内核级别缓冲整个HTTP请求。一旦一个请求体被完整接收，内核将把它发送给服务器。参见accf_http(9)手册页以获得更详细的信息。因为HTTPS请求已经被加密了，所以只使用了accf_data(9)过滤器。

Linux上的默认值是：

```
AcceptFilter http data
AcceptFilter https data
```

Linux的 `TCP_DEFER_ACCEPT` 并不支持对http请求进行缓冲。除 `none` 之外的任何值都将在监听程序上启用 `TCP_DEFER_ACCEPT` 。参见tcp(7)手册页以获得更多详情。

使用 `none` 将会为那个协议禁用接收过滤器(accept filter)。这对于像 `nntp` 这样需要服务器先发送数据的协议很有用处：

```
AcceptFilter nntp none
```

AcceptPathInfo 指令

说明	是否接受附带多余路径名信息的请求
语法	`AcceptPathInfo On|Off|Default`
默认值	`AcceptPathInfo Default`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core
兼容性	仅在 Apache 2.0.30 及以后的版本中可用

此指令决定是否接受在实际文件名(或实际目录中一个不存在的文件)后跟随多余路径名信息的请求。这个多余的路径名信息可以当作 `PATH_INFO` 环境变量传递给脚本。

比如说，假设 `/test/` 所指向的目录下只包括一个文件： `here.html` ，那么对 `/test/here.html/more` 和 `/test/nothere.html/more` 的请求都会将 `PATH_INFO` 环境变量设为" `/more` "。

`AcceptPathInfo` 指令的取值范围：

`Off`

仅当一个请求映射到一个真实存在的路径时，才会被接受。这样，如上述 `/test/here.html/more` 这样在真实文件名后跟随一个路径名的请求将会返回一个"404 NOT FOUND"错误。

`On`

只要前导路径可以映射到一个真实存在的文件，就可以接受该请求。这样，只要上述 `/test/here.html` 能够映射到一个有效的文件，那么对 `/test/here.html/more` 的请求就会被接收。

`Default`

是否接收附带多余路径名信息的请求由其对应的处理器来决定。对应普通文本的核心处理器默认会拒绝 `PATH_INFO` 。而用于伺服脚本的处理器，比如cgi-script和isapi-isa，默认会接受 `PATH_INFO` 。

`AcceptPathInfo` 指令存在的首要目的就是允许您覆盖处理器关于是否接受 `PATH_INFO` 的默认设置。这种覆盖是很必要的。比如说，当您使用了类似INCLUDES这样的过滤器来根据 `PATH_INFO` 产生内容时。核心处理器通常会拒绝这样的请求，而您就可以用下述的配置使这样的脚本成为可能：

```
<Files "mypaths.shtml">
Options +Includes
SetOutputFilter INCLUDES
AcceptPathInfo On
</Files>
```

AccessFileName 指令

说明	分布式配置文件的名字
语法	`AccessFileName filename`
默认值	`AccessFileName .htaccess`
作用域	server config, virtual host
状态	核心(C)
模块	core

如果为某个目录启用了分布式配置文件功能，那么在向客户端返回其中的文档时，服务器将在这个文档所在的各级目录中查找此配置文件。比如：

```
AccessFileName .acl
```

在返回文档 `/usr/local/web/index.html` 之前，服务器会为此指令读取 `/.acl` 、 `/usr/.acl` 、 `/usr/local/.acl` 、 `/usr/local/web/.acl` 除非此功能以被如下配置所禁用：

```
<Directory />
AllowOverride None
</Directory>
```

参见

- `AllowOverride`
- 配置文件
- .htaccess 文件

AddDefaultCharset 指令

说明	当应答内容是 `text/plain` 或 `text/html` 时，在HTTP应答头中加入的默认字符集
语法	`AddDefaultCharset On|Off|charset`
默认值	`AddDefaultCharset Off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core

当且仅当应答内容是 `text/plain` 或 `text/html` 时，此指令将会在HTTP应答头中加入的默认字符集。理论上这将覆盖在文档体中通过 `<meta>` 标签指定的字符集，但是实际的行为通常取决于用户浏览器的设置。 `AddDefaultCharset Off` 将会禁用此功能。 `AddDefaultCharset On` 将启用Apache内部的默认字符集 `iso-8859-1` 。您也可以指定使用在IANA注册过的字符集名字中的另外一个charset 。比如说：

```
AddDefaultCharset utf-8
```

`AddDefaultCharset` 只应当在如下情况下使用：所有文本资源都使用同一种确定的字符集，且分别标记他们的字符集非常麻烦。一个这样的例子是向包含动态内容的资源中添加字符集参数(比如先前遗留的CGI脚本)，这样可能会因为在输出中包含用户提供的数据而导致跨站点脚本攻击。但是请注意：更好的解决办法是修改或删除这些脚本，因为设置了默认的字符集以后将会使得浏览器的字符集自动探测功能失效。

参见

- `AddCharset`

AddOutputFilterByType 指令

说明	对特定的**MIME**类型指定输出过滤器
语法	`AddOutputFilterByType filter[;filter...] MIME-type [MIME-type] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core
兼容性	Apache 2.0.33 以后可用，但在Apache 2.1 以后反对使用

此指令根据应答的MIME类型对请求激活特定的输出过滤器。由于下面将要讨论的问题，我们反对使用这个指令。同样的功能可以通过使用 `mod_filter` 模块获得。

下例使用了由 `mod_deflate` 提供的 `DEFLATE` 过滤器。它将把所有以 `text/html` 或 `text/plain` 为标记的输出(不论静态或动态)在发送到客户端之前进行压缩。

```
AddOutputFilterByType DEFLATE text/html text/plain
```

如果您希望使用多个过滤器来处理内容，您可以用分号(;)来分隔它们的名字。并对每个过滤器使用 `AddOutputFilterByType` 指令。

下述配置将使所有标记为 `text/html` 的脚本输出首先被 `INCLUDES` 过滤器处理后再被 `DEFLATE` 过滤器处理。

```
<Location /cgi-bin/>
Options Includes
AddOutputFilterByType INCLUDES;DEFLATE text/html
</Location>
```

注意

在某些情况下，用 `AddOutputFilterByType` 来使用过滤器会遭受部分或完全的失败。比如，如果MIME类型不能确定，那么将不会有过滤器加于其上，从而使之回到 `DefaultType` 的设置。甚至当 `DefaultType` 与其相同的时候也是这样。

然而，如果您想确认对某些资源相关的内容类型确实使用了过滤器，您可以使用用诸如 `AddType` 或 `ForceType` 这样的办法。在一个(non-nph)CGI脚本中设定内容类型也很安全。

由类型决定的输出过滤器永远不会作用于来自代理的请求。

参见

- `AddOutputFilter`

- `SetOutputFilter`
- 过滤器

AllowEncodedSlashes 指令

说明	确定是否允许URL中使用经过编码的路径分割符
语法	`AllowEncodedSlashes On|Off`
默认值	`AllowEncodedSlashes Off`
作用域	server config, virtual host
状态	核心(C)
模块	core
兼容性	仅在 Apache 2.0.46 及以后的版本中可用

`AllowEncodedSlashes` 指令允许使用包含经过编码的路径分割符的URL(" `%2F` "→" `/` "或" `%5C` "→" `\` "，取决于不同的系统)。默认情况下，这些URL将被一个包含"404"(未找到)错误的应答拒绝。

`AllowEncodedSlashes` `On` 通常和 `PATH_INFO` 配合使用。

注意

允许使用经过编码的斜线(路径分割符)并不意味着解码。 `%2F` 或 `%5C` (仅仅取决于不同的系统)将会按原样出现在解码后的URL字符串中。

参见

- `AcceptPathInfo`

AllowOverride 指令

说明	确定允许存在于 `.htaccess` 文件中的指令类型
语法	`AllowOverride All|None|directive-type [directive-type] ...`
默认值	`AllowOverride All`
作用域	directory
状态	核心(C)
模块	core

当服务器发现一个 `.htaccess` 文件(由 `AccessFileName` 指定)时，它需要知道在这个文件中声明的哪些指令能覆盖在此之前指定的配置指令。

仅允许存在于<Directory>配置段

`AllowOverride` 仅在不包含正则表达式的 `<Directory>` 配置段中才是有效的。在 `<Location>` , `<DirectoryMatch>` , `<Files>` 配置段中都是无效的。

如果此指令被设置为 `None` ，那么.htaccess文件将被完全忽略。事实上，服务器根本不会读取 `.htaccess` 文件。

当此指令设置为 `All` 时，所有具有".htaccess"作用域的指令都允许出现在 `.htaccess` 文件中。

directive-type可以是下列各组指令之一：

AuthConfig

允许使用与认证授权相关的指令(`AuthDBMGroupFile` , `AuthDBMUserFile` , `AuthGroupFile` , `AuthName` , `AuthType` , `AuthUserFile` , `Require` , 等)。

FileInfo

允许使用控制文档类型的指令(`DefaultType` , `ErrorDocument` , `ForceType` , `LanguagePriority` , `SetHandler` , `SetInputFilter` , `SetOutputFilter` , `mod_mime` 中的 Add 和 *Remove* 指令等等)、控制文档元数据的指令(`Header` , `RequestHeader` , `SetEnvIf` , `SetEnvIfNoCase` , `BrowserMatch` , `CookieExpires` , `CookieDomain` , `CookieStyle` , `CookieTracking` , `CookieName`)、 `mod_rewrite` 中的指令(`RewriteEngine` , `RewriteOptions` , `RewriteBase` , `RewriteCond` , `RewriteRule`)和 `mod_actions` 中的 `Action` 指令。

Indexes

允许使用控制目录索引的指令(`AddDescription` , `AddIcon` , `AddIconByEncoding` , `AddIconByType` , `DefaultIcon` , `DirectoryIndex` , `FancyIndexing` , `HeaderName` , `IndexIgnore` , `IndexOptions` , `ReadmeName` , 等)。

Limit

允许使用控制主机访问的指令(`Allow` , `Deny` , `Order`)。

Options[=Option,...]

允许使用控制指定目录功能的指令(`Options` 和 `XBitHack`)。可以在等号后面附加一个逗号分隔的(无空格的) `Options` 选项列表，用来控制允许 `Options` 指令使用哪些选项。

例如以下指令只允许在 `.htaccess` 中使用 `AuthConfig` 和 `Indexes` 组的指令：

```
AllowOverride AuthConfig Indexes
```

不在这两组中的指令将会导致服务器产生一个内部错误。

参见

- `AccessFileName`
- 配置文件
- .htaccess文件

AuthName 指令

说明	用于HTTP认证的授权域
语法	`AuthName auth-domain`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	核心(C)
模块	core

此指令为目录的授权域设置名字。此域将发送给客户端以使用户了解应当发送哪个用户名和密码。`AuthName` 指令带有一个参数。如果域的名字中包含空格，则必须用引号引起来。它必须与 `AuthType` 和 `Require` 指令以及诸如 `AuthUserFile` 和 `AuthGroupFile` 这样的指令一起工作。

例如：

```
AuthName "Top Secret"
```

提供给 `AuthName` 的字符串将出现在大多数浏览器提供的密码对话框中。

参见

- 认证、授权、访问控制

AuthType 指令

说明	用户认证类型
语法	`AuthType Basic|Digest`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	核心(C)
模块	core

此指令选择了一个目录的用户认证类型。目前只实现了 `Basic` (`mod_auth_basic`) 和 `Digest` (`mod_auth_digest`)。

要实现认证，还必须同时与 `AuthName` 和 `Require` 指令一起使用。另外，服务器还必须包含一个认证支持模块(比如 `mod_authn_file`)和一个授权支持模块(比如 `mod_authz_user`)。

参见

- 认证、授权、访问控制

CGIMapExtension 指令

说明	定位CGI脚本解释器
语法	`CGIMapExtension cgi-path .extension`
作用域	directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core
兼容性	NetWare only

此指令用于定位Apache CGI脚本解释器。比如，" `CGIMapExtension sys:\foo.nlm .foo` "将把所有具有 `.foo` 后缀的CGI脚本文件传递给FOO解释器。

ContentDigest 指令

说明	允许生成 `Content-MD5` 应答头
语法	ContentDigest On|Off
默认值	`ContentDigest Off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Options
状态	核心(C)
模块	core

此指令遵照RFC1854和RFC2068协议的定义启用了 `Content-MD5` 应答头的生成。

MD5是一种为不定长度的数据计算出一个"消息摘要"(有时也称为"指纹")的算法。并且保证数据中的任何变化都会反应在消息摘要的变化中。

`Content-MD5` 头提供了一种端到端的针对整个消息体的信息完整性检查方法。代理或者客户端会检查此头以侦测在传输过程中，消息体是否产生了意外的改变。一个头的例子如下：

```
Content-MD5: AuLb7Dp1rqtRtxz2m9kRpA==
```

请注意，因为对每个请求都要进行消息摘要的运算(没有对其值进行缓存)，所以这会对您的服务器造成性能方面的影响。

`Content-MD5` 仅为由 `Apache核心` 伺服的文档进行发送，而对于由模块处理的文档则不予理会。比如说SSI文档、CGI脚本的输出、字节范围的应答都不包括这个头。

DefaultType 指令

说明	在服务器无法由其他方法确定内容类型时，发送的默认**MIME**内容类型
语法	`DefaultType MIME-type`
默认值	`DefaultType text/plain`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core

有时会发生这样的事：服务器会被要求提供一个文档，而这个文档的类型无法由它的MIME类型映射所决定。

服务器必须通知客户端其文档的内容类型。所以当一个未知类型出现时，将会使用 `DefaultType` 。例如：

```
DefaultType image/gif
```

这样的配置对于里面有很多gif图片而有些在文件名中缺少 `.gif` 扩展名的目录非常合适。

注意，与 `ForceType` 指令的不同之处在于：此指令仅提供了默认的mime类型。所有其它mime类型的定义，包括文件的扩展名，或其它可以标识媒体类型的方法都会覆盖此默认值。

<Directory> 指令

说明	封装一组指令，使之仅对文件空间中的某个目录及其子目录生效
语法	`<Directory directory-path> ... </Directory>`
作用域	server config, virtual host
状态	核心(C)
模块	core

`<Directory>` 和 `</Directory>` 用于封装一组指令，使之仅对某个目录及其子目录生效。任何可以在"directory"作用域中使用的指令都可以使用。Directory-path可以是一个目录的完整路径，或是包含了Unix shell匹配语法的通配符字符串。在通配符字符串中，" `?` "匹配任何单个的字符，" `*` "匹配任何字符序列。您也可以使用" `[]` "来确定字符范围。以上通配符都不能匹配"/"字符。所以 `<Directory /*/public_html>` 将无法匹配 `/home/user/public_html` ，但 `<Directory /home/*/public_html>` 能够正确匹配。比如说：

```
<Directory /usr/local/httpd/htdocs>

Options Indexes FollowSymLinks
</Directory>
```

使用directory-path参数的时候要注意：它们必须与Apache用于访问文件的文件系统路径保持一致。赋予特定 `<Directory>` 的指令将无法对通过不同路径指向的同一个目录文件生效，比如说通过另外一个符号连接生成的路径。

扩展的正则表达式也可以通过附加一个" `~` "字符来使用。比如说：

```
<Directory ~ "^/www/(.+/)*[0-9]{3}">
```

将匹配 `/www/` 下所有由3个数字组成的目录。

如果有多个(非正则表达式) `<Directory>` 配置段符合包含某文档的目录(或其父目录)，那么指令将以短目录优先的规则进行应用。并包含.htaccess文件中的指令。比如说在

```
<Directory />

AllowOverride None
</Directory>

<Directory /home/>

AllowOverride FileInfo
</Directory>
```

中，访问文档 `/home/web/dir/doc.html` 的步骤如下：

- 应用指令 `AllowOverride None` (禁用 `.htaccess` 文件)。
- 应用指令 `AllowOverride FileInfo` (针对 `/home` 目录)。
- 按顺序应用所有 `/home/.htaccess` 、 `/home/web/.htaccess` 、 `/home/web/dir/.htaccess` 中的 `FileInfo` 组指令。

正则表达式将在所有普通配置段之后予以考虑。所有的正则表达式将根据它们出现在配置文件中的顺序进行应用。比如说，以下配置：

```
<Directory ~ abc$>

# ......
</Directory>
```

正则表达式配置段将在所有普通的 `<Directory>` 和 `.htaccess` 文件应用之后才予以考虑。所以正则表达式将匹配 `/home/abc/public_html/abc` 并予以应用。

请注意：**Apache**对 **`<Directory />`** 的默认访问权限为" **`Allow from All`** "。这意味着**Apache**将伺服任何通过**URL**映射的文件。我们建议您将这个配置做如下屏蔽：

```
<Directory />

Order Deny,Allow

  Deny from All
</Directory>
```

然后在您想要使之被访问的目录中覆盖此配置。参阅安全提示以获取更多详情。

一般来说 `<Directory>` 指令只会出现在 `httpd.conf` 文件中，但它们也可能出现在任何其它配置文件中。 `<Directory>` 指令不可被嵌套使用，也不能出现在 `<Limit>` 或 `<LimitExcept>` 配置段中。

参见

- <Directory>、<Location>、<Files>配置段是如何工作的中包含了当接受一个请求时，这些不同的配置段是如何组合工作的相关解释。

<DirectoryMatch> 指令

说明	封装一些指令并作用于文件系统中匹配正则表达式的所有目录及其子目录
语法	`<DirectoryMatch regex> ... </DirectoryMatch>`
作用域	server config, virtual host
状态	核心(C)
模块	core

`<DirectoryMatch>` 和 `</DirectoryMatch>` 用于封装一组指令。与 `<Directory>` 类似，此指令将仅作用于指定名字的目录及其子目录。然而，它可以接受一个正则表达式作为参数。比如说：

```
<DirectoryMatch "^/www/(.+/)*[0-9]{3}">
```

将匹配 `/www/` 下所有由3个数字组成的目录。

参见

- `<Directory>` 获取如何在普通的 `<Directory>` 中使用正则表达式的描述。
- <Directory>、<Location>、<Files>配置段是如何工作的中包含了当接受一个请求时，这些不同的配置段是如何组合工作的相关解释。

DocumentRoot 指令

说明	组成网络上可见的主文档树的根目录
语法	`DocumentRoot directory-path`
默认值	`DocumentRoot /usr/local/apache2/htdocs`
作用域	server config, virtual host
状态	核心(C)
模块	core

此指令设置了 `httpd` 伺服的目录。在没有使用类似 `Alias` 这样的指令的情况下，服务器会将请求中的URL附加到 `DocumentRoot` 后面以构成指向文档的路径。比如说：

```
DocumentRoot /usr/web
```

于是对 `http://www.my.host.com/index.html` 的访问就会指向 `/usr/web/index.html` 。如果directory-path不是绝对路径，则被假定为是相对于 `ServerRoot` 的路径。

指定 `DocumentRoot` 时不应包括最后的"/"。

参见

- 从URL到文件系统的映射

EnableMMAP 指令

说明	在递送中使用内存映射(memory-mapping)来读取文件
语法	`EnableMMAP On|Off`
默认值	`EnableMMAP On`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core

此指令指示 `httpd` 在递送中如果需要读取一个文件的内容，它是否可以使用内存映射。当处理一个需要访问文件中的数据的请求时，比如说当递送一个使用 `mod_include` 进行服务器端分析的文件时，如果操作系统支持，Apache将默认使用内存映射。

这种内存映射有时会带来性能的提高，但在某些情况下，您可能会需要禁用内存映射以避免一些操作系统的问题：

- 在一些多处理器的系统上，内存映射会减低一些 `httpd` 的性能。
- 在挂载了NFS的 `DocumentRoot` 上，若已经将一个文件进行了内存映射，则删除或截断这个文件会造成 `httpd` 因为分段故障而崩溃。

在可能遇到这些问题的服务器配置过程中，您应当使用下面的命令来禁用内存映射：

```
EnableMMAP Off
```

对于挂载了NFS的文件夹，可以单独指定禁用内存映射：

```
<Directory "/path-to-nfs-files">
EnableMMAP Off
</Directory>
```

EnableSendfile 指令

说明	使用操作系统内核的sendfile支持来将文件发送到客户端
语法	`EnableSendfile On|Off`
默认值	`EnableSendfile On`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core
兼容性	仅在 Apache 2.0.44 及以后的版本中可用

这个指令控制 `httpd` 是否可以使用操作系统内核的sendfile支持来将文件发送到客户端。默认情况下，当处理一个请求并不需要访问文件内部的数据时(比如发送一个静态的文件内容)，如果操作系统支持，Apache将使用sendfile将文件内容直接发送到客户端而并不读取文件。

这个sendfile机制避免了分开的读和写操作以及缓冲区分配，但是在一些平台或者一些文件系统上，最好禁止这个特性来避免一些问题：

- 一些平台可能会有编译系统检测不到的有缺陷的sendfile支持，特别是将在其他平台上使用交叉编译得到的二进制文件运行于当前对sendfile支持有缺陷的平台时。
- 在Linux上启用IPv6时，使用sendfile将会触发某些网卡上的TCP校验和卸载bug。
- 当Linux运行在Itanium处理器上的时候，sendfile可能无法处理大于2GB的文件。
- 对于一个通过网络挂载了NFS文件系统的 `DocumentRoot` (比如：NFS或SMB)，内核可能无法可靠的通过自己的缓冲区服务于网络文件。

如果出现以上情况，你应当禁用sendfile ：

```
EnableSendfile Off
```

针对NFS或SMB，这个指令可以被针对目录的设置覆盖：

```
<Directory "/path-to-nfs-files">
EnableSendfile Off
</Directory>
```

ErrorDocument 指令

说明	当遇到错误的时候服务器将给客户端什么样的应答
语法	`ErrorDocument error-code document`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core
兼容性	在Apache2.0中引用文字的语法发生了变化

当遇到问题或错误的时候，Apache能被配置为进行以下四种处理之一：

1. 输出一个简单生硬的错误代码信息
2. 输出一个经过定制的信息
3. 重定向到一个本地的URL-path来处理这个问题(错误)
4. 重定向到一个外部的URL来处理这个问题(错误)

默认会采取第1种方法，而第2-4种方法可以使用 `ErrorDocument` 指令后面跟随一个HTTP应答代码和一个URL或信息来进行配置。Apache有时会额外提供一些信息来描述所发生的问题/错误。

URL可以由一个斜杠(/)开头来指示一个本地URL(相对于 `DocumentRoot`)，或是提供一个能被客户端解释的完整的URL。此外还能提供一个可以被浏览器显示的消息。比如：

```
ErrorDocument 500 http://foo.example.com/cgi-bin/tester

ErrorDocument 404 /cgi-bin/bad_urls.pl

ErrorDocument 401 /subscription_info.html

ErrorDocument 403 "Sorry can't allow you access today"
```

另外，特殊的" `default` "值可以被用来指定使用Apache内置的、简单的硬编码消息。当不需要这个定制特性的时候，可以用" `default` "恢复Apache内置的、简单的硬编码消息，否则将继承一个已有的 `ErrorDocument` 。

```
ErrorDocument 404 /cgi-bin/bad_urls.pl

<Directory /web/docs>

ErrorDocument 404 default
</Directory>
```

请注意，如果您为 `ErrorDocument` 指定了一个外部的URL(比如说，任何在开头指示了类似" `http` "这样的访问方法的字符串)，Apache将会向客户端发送一个重定向指令来告诉它在哪里找到这个文档，哪怕这个文档最后还是在这个服务器上。这里面包含着一些暗示：最重要的就是客户端无法接收到原始的错误状态代码，取而代之的是一个重定向状态代码。这将

会使一些用状态代码来判断一个URL是否有效的web机器人或其它客户端产生误解。另外，如果您在" `ErrorDocument 401` "中使用了外部URL，客户端将不会提示用户输入密码，因为它根本没收到这样一个401的状态代码。所以，如果您想使用" **`ErrorDocument 401`** "指令，就必须指向一个本地的文档。

Microsoft Internet Explorer (MSIE)在服务器端产生的错误信息"很小"的时候会忽略它们而用自己"友好的"错误信息进行取代。这个大小的阈值根据错误类型而不同。但一般来说，如果您的错误信息的大小在512 byte以上，MSIE就会显示这些服务器端产生的错误文档而不会屏蔽它们。您可以在微软知识库的文章Q294807中获取更多信息。

虽然大多数错误信息可以被改写，但是在有些情况下，将仍然使用某些内置的错误信息而不管 `ErrorDocument` 如何设置。特别是在检测到一个"畸形"请求的情况下，正常的请求处理过程将会被立即中断，并且立即返回一个内置的错误信息。这是为了防止某些不良请求可能导致的安全问题。

在2.0版以前，信息前面会用一个不配对的双引号作为前导标志。

参见

- 定制个性化应答文档

ErrorLog 指令

说明	存放错误日志的位置
语法	`ErrorLog file-path|syslog[:facility]`
默认值	`ErrorLog logs/error_log (Unix) ErrorLog logs/error.log (Windows 和 OS/2)`
作用域	server config, virtual host
状态	核心(C)
模块	core

`ErrorLog` 指令指定了当服务器遇到错误时记录错误日志的文件。如果file-path不是一个以斜杠(/)开头的绝对路径，那么将被认为是一个相对于 `ServerRoot` 的相对路径。

示例

```
ErrorLog /var/log/httpd/error_log
```

如果file-path以一个管道符号(|)开头，那么会为它指定一个命令来处理错误日志。

示例

```
ErrorLog "|/usr/local/bin/httpd_errors"
```

如果系统支持，使用" `syslog` "替代文件名将通过syslogd(8)来记载日志。默认将使用系统日志机制 `local7` ，但您可以用" `syslog:facility` "语法来覆盖这个设置，其中，facility的取值为syslog(1)中记载的任何一个名字。

示例

```
ErrorLog syslog:user
```

安全提示：参阅安全提示文档获得关于为什么当记录日志文件的目录对于启动服务器以外的用户可写时会对您的服务器构成安全威胁。

注意

当在非Unix平台上输入文件路径的时候，路径分隔符必须统一使用正斜线(/)，即使那个平台本身使用反斜线()。

参见

- `LogLevel`
- Apache日志文件

FileETag 指令

说明	用以创建ETag应答头的文件的属性
语法	`FileETag component ...`
默认值	`FileETag INode MTime Size`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core

`FileETag` 指令配置了当文档是基于一个文件时用以创建 `ETag` (实体标签)应答头的文件的属性(`ETag` 的值用于进行缓冲管理以节约网络带宽)。在Apache1.3.22及以前， `ETag` 的值总是由文件的inode(索引节点)、大小、最后修改时间决定。 `FileETag` 指令可以让您选择(如果您想进行选择)这其中哪些要素将被使用。主要关键字如下：

INode

文件的索引节点(inode)数

MTime

文件的最后修改日期及时间

Size

文件的字节数

All

所有存在的域，等价于：

```
FileETag INode MTime Size
```

None

如果一个文档是基于文件的，则不在应答中包含任何 `ETag` 头

可以在 `INode` ， `MTime` ， `Size` 前加上" `+` "或" `-` "以改变由上层继承下来的默认值。任何没有上述前缀的关键字将立刻完全取消继承下来的设置。

如果一个目录的配置包含了" `FileETag INode MTime Size` "而其一个子目录包含了" `FileETag -INode` "那么这个子目录的设置(并会被其下任何没有进行覆盖的子目录继承)将等价于" `FileETag MTime Size` "。

<Files> 指令

说明	包含作用于匹配指定文件名的指令
语法	`<Files filename> ... </Files>`
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

`<Files>` 指令提供了基于文件名的访问控制，类似于 `<Directory>` 和 `<Location>` 指令。它将配对一个 `</Files>` 指令。在此配置段中定义的指令将作用于其基本名称(不是完整的路径)与指定的文件名相符的对象。 `<Files>` 段将根据它们在配置文件中出现的顺序被处理：在 `<Directory>` 段和 `.htaccess` 文件被处理之后，但在 `<Location>` 段之前。请注意： `<Files>` 能嵌入到 `<Directory>` 段中以限制它们作用的文件系统范围。

filename参数应当是一个文件名或是一个包含通配符的字符串，其中" ? "匹配任何单个字符，" * "匹配任何字符串序列。在" ~ "字符之后同样可以使用正则表达式。比如：

```
<Files ~ "\.(gif|jpe?g|png)$">
```

将匹配绝大部分常见的因特网图象格式。然而在Apache1.3及其后继版本中，更推荐使用 <FilesMatch> 指令。

请注意与 <Directory> 和 <Location> 配置段不同的是： <Files> 配置段可用于 .htaccess 文件当中。这将允许用户在文件层面上控制对它们自己文件的访问。

参见

- <Directory>、<Location>、<Files>配置段是如何工作的中包含了当接受一个请求时，这些不同的配置段是如何组合工作的相关解释。

<FilesMatch> 指令

说明	包含作用于与正则表达式匹配的文件名的指令
语法	<FilesMatch regex> ... </FilesMatch>
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

<FilesMatch> 指令就像 <Files> 指令一样提供了针对文件名的访问控制。然而，它使用的是正则表达式。比如说：

```
<FilesMatch "\.(gif|jpe?g|png)$">
```

将匹配最常见的internet图形文件格式。

参见

- <Directory>、<Location>、<Files>配置段是如何工作的中包含了当接受一个请求时，这些不同的配置段是如何组合工作的相关解释。

ForceType 指令

说明	强制所有匹配的文件被作为指定的MIME类型进行伺服
语法	`ForceType MIME-type|None`
作用域	directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core
兼容性	Apache 2.0之后从其它模块移动到核心中

当此指令放入 `.htaccess` 文件或 `<Directory>` 或 `<Location>` 或 `<Files>` 配置段时，此指令强制所有匹配的文件被当作在MIME-type中指定的Content-Type来伺服。比如说，如果您有一个包含大量GIF文件的目录，可您又不想全都为它们加上" `.gif` "扩展名的话，您可以这样做：

```
ForceType image/gif
```

请注意：与 `DefaultType` 指令不同，此指令将覆盖所有的mime类型关联，包括标识文件类型的扩展名。

你可以通过使用" `None` "覆盖任何 `ForceType` 设置：

```
# 强制所有文件为 image/gif:

<Location /images>

ForceType image/gif
</Location>

# 但是正常的mime类型关联在这里:

<Location /images/mixed>

ForceType None
</Location>
```

HostnameLookups 指令

说明	启用对客户端IP的DNS查找
语法	`HostnameLookups On|Off|Double`
默认值	`HostnameLookups Off`
作用域	server config, virtual host, directory
状态	核心(C)
模块	core

此指令启用了DNS查询，使得主机名能被记入日志(并用 REMOTE_HOST 变量传递给CGI/SSI)。参数 Double 指定进行一次双向DNS查询。也就是说在一次反向查询之后，再对返回的结果进行一次正向查询。在正向查询结果中至少应该有一个ip地址与初始的地址相符。(在"tcpwrappers"中的术语是 PARANOID)

不论此处如何设置，当您使用 mod_authz_host 来根据主机名控制访问的时候，就会执行一次双向查询。这对安全来说非常必要。请注意如果您没有设置" HostnameLookups Double "，这种双向查询的结果不是自动生成的。比如说：如果仅仅设置了" HostnameLookups On "而且请求是针对一个根据主机名做了限制的对象，不论双向查询是否失败，CGI还是会把单向查询的结果用 REMOTE_HOST 来传送。

默认值设置为 Off 是为了那些不需要进行反向查询的站点节约网络带宽考虑的。这对最终用户也是有益的，因为这样他们就不用忍受查询造成的延迟了。高访问量的网站应该将此指令设置为 Off 因为DNS查询会造成明显的时间消耗。在 bin 目录下的 logresolve 工具可以在离线的情况下对已经记入日志的IP地址进行主机名的查询。

<IfDefine> 指令

说明	封装一组只有在启动时当测试结果为真时才生效的指令
语法	<IfDefine [!]parameter-name> ... </IfDefine>
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

<IfDefine test>...</IfDefine> 配置段用于包含有条件的指令。 <IfDefine> 配置段中的指令仅当test结果为真时才进行处理。如果test为假。此配置段中的指令将会被忽略。

<IfDefine> 配置段中的test可以为以下两种形式之一：

- parameter-name
- ! parameter-name

在第一种情况下，仅当parameter-name已经定义的情况下才对开始和结束标记之间的指令进行处理。第二种情况则截然相反。仅当parameter-name没有定义的情况下才进行指令的处理。

parameter-name是在服务启动时，通过 httpd 命令行的 -Dparameter 这样的形式指定的。

<IfDefine> 配置段是可以嵌套的，从而可以实现简单的多参数测试。比如说：

```
        httpd -DReverseProxy ...

        # httpd.conf

        <IfDefine ReverseProxy>

   LoadModule rewrite_module modules/mod_rewrite.so

          LoadModule proxy_module   modules/libproxy.so
        </IfDefine>
```

<IfModule> 指令

说明	封装指令并根据指定的模块是否启用为条件而决定是否进行处理
语法	<IfModule [!]module-file|module-identifier> ... </IfModule>
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core
兼容性	module-identifier仅在 Apache 2.1 及以后的版本中可用

<IfModule test>...</IfModule> 配置段用于封装根据指定的模块是否启用而决定是否生效的指令。在 <IfModule> 配置段中的指令仅当test为真的时候才进行处理。如果test为假，所有其间的指令都将被忽略。

<IfModule> 段中的test可以为以下两种方式之一：

- module
- !module

在第一种情况下，起始和结束标记之间的指令仅当module被载入后才被执行。此模块可以为编译时静态链接进核心的模块或是使用 LoadModule 指令动态载入的模块。第二种情况则相反，仅当module没有载入时才进行指令的处理。

module可以是模块的标识符或者是编译模块时的文件名。比如， rewrite_module 就是一个模块标识符，而 mod_rewrite.c 则是编译模块时的文件名。如果模块包含多个源代码文件，您应当使用包含 STANDARD20_MODULE_STUFF 字符串的那个。

<IfModule> 配置段是可以嵌套的，从而可以实现简单的多模块测试。

此配置段主要用于需要根据某个特定的模块是否存在来决定是否使用某些配置的时候。指令一般都放在 <IfModule> 配置段中。

Include 指令

说明	在服务器配置文件中包含其它配置文件
语法	`Include file-path|directory-path`
作用域	server config, virtual host, directory
状态	核心(C)
模块	core
兼容性	通配符仅在 Apache 2.0.41 及以后的版本中可用

这个指令允许在服务器配置文件中加入其它配置文件。

Shell风格(`fnmatch()`)的通配符可以用于按照字母顺序一次包含多个文件。另外，如果 `Include` 指向了一个目录而不是一个文件，Apache将读入该目录及其子目录下的所有文件，并依照字母顺序将这些文件作为配置文件进行解析。但是并不推荐这么做，因为偶尔会有临时文件在这个目录中生成，从而导致 `httpd` 启动失败。

文件的路径可以是一个完整的绝对路径(以一个斜杠开头)：

```
Include /usr/local/apache2/conf/ssl.conf
Include /usr/local/apache2/conf/vhosts/*.conf
```

或是相对于 `ServerRoot` 目录的相对路径：

```
Include conf/ssl.conf
Include conf/vhosts/*.conf
```

请确保包含的目录中不包含任何诸如编辑器临时文件等引起误导的文件，因为Apache会尝试读取它们并把其中的内容作为配置指令来处理，这样可能会导致启动过程的失败。运行 `apachectl configtest` 将会把配置检查时所使用的所有文件列出来以供参考。这将有助于检验配置中是否仅包含了您所希望出现那些文件。

```
root@host# apachectl configtest
Processing config file: /usr/local/apache2/conf/ssl.conf
Processing config file: /usr/local/apache2/conf/vhosts/vhost1.conf
Processing config file: /usr/local/apache2/conf/vhosts/vhost2.conf
Syntax OK
```

参见

- `apachectl`

KeepAlive 指令

说明	启用HTTP持久链接
语法	`KeepAlive On|Off`
默认值	`KeepAlive On`
作用域	server config, virtual host
状态	核心(C)
模块	core

Keep-Alive扩展自HTTP/1.0和HTTP/1.1的持久链接特性。提供了长效的HTTP会话，用以在同一个TCP连接中进行多次请求。在某些情况下，这样的方式会对包含大量图片的HTML文档造成的延时起到50%的加速作用。在Apache1.2版本以后，您可以设置 `KeepAlive On` 以启用持久链接。

对于HTTP/1.0的客户端来说，仅当客户端指定使用的时候才会使用持久链接连接。此外，仅当能够预先知道传输的内容长度时，才会与HTTP/1.0的客户端建立持久链接连接。这意味着那些长度不定的内容，诸如CGI输出、SSI页面、以及服务器端生成的目录列表等内容一般来说将无法使用与HTTP/1.0客户端建立的持久链接连接。而对于HTTP/1.1的客户端来说，如果没有进行特殊指定，持久将是默认的连接方式。如果客户端进行了请求，将使用分块编码以解决在持久链接里发送未知长度内容的问题。

参见

- `MaxKeepAliveRequests`

KeepAliveTimeout 指令

说明	持久链接中服务器在两次请求之间等待的秒数
语法	`KeepAliveTimeout seconds`
默认值	`KeepAliveTimeout 5`
作用域	server config, virtual host
状态	核心(C)
模块	core

Apache在关闭持久连接前等待下一个请求的秒数。一旦收到一个请求，超时值将会被设置为 `Timeout` 指令指定的秒数。

对于高负荷服务器来说， `KeepAliveTimeout` 值较大会导致一些性能方面的问题：超时值越大，与空闲客户端保持连接的进程就越多。

<Limit> 指令

说明	仅对指定的HTTP方法进行访问控制
语法	`<Limit method [method] ... > ... </Limit>`
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

访问控制一般来说是对所有的访问方法都生效的，这也是我们普遍希望达到的效果。一般情况下，访问控制指令不应该放入 **`<Limit>`** 段中。

`<Limit>` 指令的目的是限制访问控制的效果使其仅作用于某些HTTP方法。对于其它方法， `<Limit>` 括号中的访问限制将不起任何作用。下例中的访问控制仅作用于 `POST` ， `PUT` ， `DELETE` 方法，其它方法不受任何影响：

```
     <Limit POST PUT DELETE>

Require valid-user
     </Limit>
```

列出的方法名可为下列的一个或多个： `GET` ， `POST` ， `PUT` ， `DELETE` ， `CONNECT` ， `OPTIONS` ， `PATCH` ， `PROPFIND` ， `PROPPATCH` ， `MKCOL` ， `COPY` ， `MOVE` ， `LOCK` ， `UNLOCK` 。方法名是大小写敏感的。如果对 `GET` 进行了定义，它会同时作用于 `HEAD` 请求。 `TRACE` 方法不能被限制。

应当总是优先使用 `<LimitExcept>` 段来限制访问，而不是 `<Limit>` 段。因为 `<LimitExcept>` 段能够防范所有HTTP方法。

<LimitExcept> 指令

说明	对除了指定方法以外的所有HTTP方法进行访问控制
语法	`<LimitExcept method [method] ... > ... </LimitExcept>`
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

`<LimitExcept>` 和 `</LimitExcept>` 用于封装一组访问控制指令，并将其作用于所有没有在参数中标出的HTTP方法。也就是说，与 `<Limit>` 相反，它用于控制标准的和非标准以及无法辨识的方法。

比如：

```
    <LimitExcept POST GET>
Require valid-user
    </LimitExcept>
```

LimitInternalRecursion 指令

说明	指定内部重定向和嵌套子请求的最大数量
语法	`LimitInternalRecursion number [number]`
默认值	`LimitInternalRecursion 10`
作用域	server config, virtual host
状态	核心(C)
模块	core
兼容性	仅在 Apache 2.0.47 及以后的版本中可用

例如，当使用 `Action` 指令内部重定向原始请求到一个CGI脚本时，一个内部重定向将会发生。子请求是Apache的一个用于找到如果一个URI被请求时将会发生什么的机制。例如， `mod_dir` 使用子请求来寻找那些根据 `DirectoryIndex` 指令应当被列出的文件。

`LimitInternalRecursion` 可以防止服务器进入一个内部重定向或者子请求的死循环而崩溃。这样的死循环通常由错误的配置引起。

这个指令存储了两个不同的限制，这两个限制是基于每个单独的请求进行计算的。第一个number限制了内部重定向链的最大长度(一个接一个)。第二个number限制了子请求的最大嵌套层数。如果你只指定了一个number ，那么将会被同时应用于这两个限制。

示例

```
    LimitInternalRecursion 5
```

LimitRequestBody 指令

说明	限制客户端发送的HTTP请求体的最大字节长度
语法	`LimitRequestBody bytes`
默认值	`LimitRequestBody 0`
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

bytes在0(意味着无限制)到2147483647(2GB)间限制了请求体所允许的字节数。

`LimitRequestBody` 可以让用户在其作用范围内(整个服务器、特定目录、特定文件、特定位置)设置一个允许客户端发送的HTTP请求体的最大字节长度的限制。如果客户端的请求超出了这个限制，服务器会回应一个错误而不是伺服这个请求。一个普通请求的信息体在很大程度上取决于资源的自然属性和这个资源允许的方法。CGI脚本经常用消息体把表单的信息传递给服务器。使用 `PUT` 方法至少会需要与服务器期望从这个资源得到的信息量差不多大小的值。

此指令给了服务器管理员更大的可控性以控制客户端不正常的请求行为。这有助于避免某些形式的拒绝服务攻击。

比如，如果您允许文件上传到某个位置，而且希望能将上传文件的大小设置为100K，您可以使用下面的指令：

```
LimitRequestBody 102400
```

LimitRequestFields 指令

说明	限制接受客户端请求中HTTP请求头域的数量
语法	`LimitRequestFields number`
默认值	`LimitRequestFields 100`
作用域	server config
状态	核心(C)
模块	core

Number是一个0(意味着不限)到32767之间的整数。默认值为编译时的常量 `DEFAULT_LIMIT_REQUEST_FIELDS` (发布值为100)。

`LimitRequestFields` 指令允许服务器管理员修改在一个HTTP请求中的请求头域的数量限制。服务器需要此值大于一个普通客户端请求中包含头域的数量。一个客户端请求头域的数量很少大于20，但根据客户端的不同这个数字有很大的差别，经常取决于用户配置他们的浏览器

扩展以支持更详细的内容协商。可选的HTTP扩展经常使用请求头域来实现。

这个指令给了服务器管理员更大的可控性以控制客户端不正常的请求行为。这有助于避免某些形式的拒绝服务攻击。如果正常使用的客户端得到了服务器的错误应答，指出其在请求中发送了过多的头域，您应该适当的增大此值。

例如：

```
LimitRequestFields 50
```

LimitRequestFieldSize 指令

说明	限制客户端发送的请求头的字节数
语法	`LimitRequestFieldsize bytes`
默认值	`LimitRequestFieldsize 8190`
作用域	server config
状态	核心(C)
模块	core

bytes指定了HTTP请求头允许的字节大小。

`LimitRequestFieldSize` 指令允许服务器管理员增加或减少HTTP请求头域大小的限制。一般来说，服务器需要此值足够大，以适应普通客户端的任何请求的头域大小。一个普通头域的大小对于不同的客户端来说是有很大差别的，一般与用户配置他们的浏览器以支持更多的内容协议密切相关。SPNEGO的认证头最大可能达到12392字节。

这个指令给了服务器管理员更大的可控性以控制客户端不正常的请求行为。这有助于避免某些形式的拒绝服务攻击。

举例如下：

```
LimitRequestFieldSize 4094
```

一般情况下，请不要改变这个设置，而是保持其默认设置。

LimitRequestLine 指令

说明	限制接受客户端发送的HTTP请求行的字节数
语法	`LimitRequestLine bytes`
默认值	`LimitRequestLine 8190`
作用域	server config
状态	核心(C)
模块	core

bytes将设置HTTP请求行的字节数限制。

`LimitRequestLine` 指令允许服务器管理员增加或减少客户端HTTP请求行允许大小的限制。因为请求行包括HTTP方法、URI、协议版本，所以 `LimitRequestLine` 指令会限制请求URI的长度。服务器会需要这个值足够大以装载它所有的资源名，包括可能在 `GET` 请求中所传递的查询部分的所有信息。

这个指令给了服务器管理员更大的可控性以控制客户端不正常的请求行为。这有助于避免某些形式的拒绝服务攻击。

举例如下：

```
LimitRequestLine 4094
```

一般情况下，不需要改变此设置的默认值。

LimitXMLRequestBody 指令

说明	限制基于XML的请求体的大小
语法	`LimitXMLRequestBody bytes`
默认值	`LimitXMLRequestBody 1000000`
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

限制基于XML的请求体大小的最大字节数，" `0` "将禁用这一检查。

比如：

```
LimitXMLRequestBody 0
```

<Location> 指令

说明	将封装的指令作用于匹配的URL
语法	`<Location URL-path|URL> ... </Location>`
作用域	server config, virtual host
状态	核心(C)
模块	core

`<Location>` 提供了基于URL的访问控制。与 `<Directory>` 指令类似，它也会启用一个以 `</Location>` 结尾的配置段。 `<Location>` 配置段的处理位于 `<Directory>` ， `.htaccess` ， `<Files>` 之后，并依照在配置文件中出现的顺序进行处理。

`<Location>` 配置段完全独立于文件系统之外操作。这有几个重要的后果。最重要的是 `<Location>` 不能用于针对文件系统的访问控制。因为可能会有几个不同的URL指向文件系统中的同一个文件，所以这样的控制常常会被很容易的绕过。

何时使用 `<Location>` ？

使用 `<Location>` 来将指令应用于独立于文件系统之外的内容。文件系统之内的内容请使用 `<Directory>` 和 `<Files>` 指令。不过一个例外是 `<Location />` ，它可以方便的作用于所用URL。

对所有的原始(非代理)请求来说，匹配的URL应该是具有" `/path/` "形式的URL路径。不包括访问方法、主机名、端口或查询字符串等。对于代理的请求，匹配的URL必须为" `scheme://servername/path` "的形式，而且必须包括前缀。

URL可以用一个通配符字符串来进行通配符的处理。" `?` "匹配任何单个的字符，而" `*` "匹配所有字符序列。

也可以附加" `~` "字符来表示使用正则表达式。例如：

```
<Location ~ "/(extra|special)/data">
```

将匹配所有包含字符串" `/extra/data` "或" `/special/data` "的URL。在Apache1.3及其后续版本中，加入了一个新的推荐使用的 `<LocationMatch>` 指令，其功能与 `<Location>` 的正则表达式版本相同。

`<Location>` 的功能在与 `SetHandler` 指令联用时能发挥最大效能。比如启用状态请求，但仅对来自 `foo.com` 的用户起效，您可以这样使用：

```
      <Location /status>
  SetHandler server-status
        Order Deny,Allow
        Deny from all
        Allow from .foo.com
      </Location>
```

请注意"/"(斜线)

斜线字符根据它在URL中出现的位置不同有着特殊的意义。大家可能都已经习惯在文件系统中，多个连续的斜线会被作为单一的斜线处理(例如" `/home///foo` "与" `/home/foo` "相同)。但在URL里面，这样是行不通的。 `<LocationMatch>` 指令和正则表达式版本的 `<Location>` 要求您明确使用多重斜线。比如： `<LocationMatch ^/abc>` 将匹配请求" `/abc` "但不会匹配请求" `//abc` "。而非正则表达式版本的 `<Location>` 指令在用于代理请求时，也有类似表现。但当非正则表达式版本的 `<Location>` 作用于非代理请求时，它会将多个毗邻的斜线认作单个斜线。比如，如果您指定了 `<Location /abc/def>` 而请求是指向" `/abc//def` "的，那么它们就是匹配的。

参见

- <Directory>、<Location>、<Files>配置段是如何工作的中包含了当接受一个请求时，这些不同的配置段是如何组合工作的相关解释。

<LocationMatch> 指令

说明	将封装的指令作用于正则表达式匹配的**URL**
语法	`<LocationMatch regex> ... </LocationMatch>`
作用域	server config, virtual host
状态	核心(C)
模块	core

`<LocationMatch>` 和 `<Location>` 指令相同，提供了基于URL的访问控制。但它使用正则表达式作为参数，而不是简单字符串。比如：

```
      <LocationMatch "/(extra|special)/data">
```

将匹配包含子串" `/extra/data` "或" `/special/data` "的URL。

参见

- <Directory>、<Location>、<Files>配置段是如何工作的中包含了当接受一个请求时，这些不同的配置段是如何组合工作的相关解释。

LogLevel 指令

说明	控制错误日志的详细程度
语法	`LogLevel level`
默认值	`LogLevel warn`
作用域	server config, virtual host
状态	核心(C)
模块	core

`LogLevel` 用于调整记录在错误日志中的信息的详细程度。(参见 `ErrorLog` 指令)。可以选择下列level，依照重要性降序排列：

Level	描述	例子
`emerg`	紧急(系统无法使用)	"Child cannot open lock file. Exiting"
`alert`	必须立即采取措施	"getpwuid: couldn't determine user name from uid"
`crit`	致命情况	"socket: Failed to get a socket, exiting child"
`error`	错误情况	"Premature end of script headers"
`warn`	警告情况	"child process 1234 did not exit, sending another SIGHUP"
`notice`	一般重要情况	"httpd: caught SIGBUS, attempting to dump core in ..."
`info`	普通信息	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."
`debug`	调试信息	"Opening config file ..."

当指定了某个级别时，所有级别高于它的信息也会被同时记录。比如，指定 `LogLevel info` ，则所有 `notice` 和 `warn` 级别的信息也会被记录。

建议至少使用 `crit` 级别。

示例如下：

```
LogLevel notice
```

注意

当错误日志是一个单独分开的正式文件的时候， `notice` 级别的消息总是会被记录下来，而不能被屏蔽。但是，当使用 `syslog` 来记录时就没有这个问题。

MaxKeepAliveRequests 指令

说明	一个持久链接中允许的最大请求数量
语法	`MaxKeepAliveRequests number`
默认值	`MaxKeepAliveRequests 100`
作用域	server config, virtual host
状态	核心(C)
模块	core

`MaxKeepAliveRequests` 指令限制了当启用 `KeepAlive` 时，每个连接允许的请求数量。如果将此值设为" `0` "，将不限制请求的数目。我们建议最好将此值设为一个比较大的值，以确保最优的服务器性能。

例如：

```
MaxKeepAliveRequests 500
```

NameVirtualHost 指令

说明	为一个基于域名的虚拟主机指定一个IP地址(和端口)
语法	`NameVirtualHost addr[:port]`
作用域	server config
状态	核心(C)
模块	core

如果您要配置基于域名的虚拟主机， `NameVirtualHost` 指令就是您必须的指令之一。

尽管addr参数可以使用主机名，但建议您还是使用IP地址。比如：

```
NameVirtualHost 111.22.33.44
```

使用 `NameVirtualHost` 指令，您可以指定一个基于域名的虚拟主机将使用哪个IP地址来接受请求。在一个防火墙或是其它代理接受了请求并把它转到服务器所在的另外一个IP地址上的情况下，您必须指定伺服请求的机器物理界面上的IP地址。如果您对于多个地址使用了多个基于域名的虚拟主机，您应该为每个地址使用这个指令。

注意

"主服务器"和任何其它" `_default_` "服务器都不会伺服发送到 `NameVirtualHost` IP地址的请求。(除非您指定了 `NameVirtualHost` ，但没有为这个地址指定任何 `VirtualHost`)。

另外，您还可以为您使用的基于域名的虚拟主机指定一个端口号。比如：

```
NameVirtualHost 111.22.33.44:8080
```

IPv6地址必须封装在一对方括号内，如下例所示：

```
NameVirtualHost [2001:db8::a00:20ff:fea7:ccea]:8080
```

为了接受所有界面的请求，您可以使用" `*` "：

```
NameVirtualHost *
```

`<VirtualHost>` 指令的参数

请注意， `<VirtualHost>` 指令的参数必须与 `NameVirtualHost` 指令的参数完全匹配。

```
NameVirtualHost 1.2.3.4
<VirtualHost 1.2.3.4>
# ...
</VirtualHost>
```

参见

- 虚拟主机文档

Options 指令

说明	配置在特定目录中可以使用哪些特性
语法	`Options [+|-]option [[+|-]option] ...`
默认值	`Options All`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Options
状态	核心(C)
模块	core

`Options` 指令控制了在特定目录中将使用哪些服务器特性。

option可以为 `None` ，在这种情况下，将不启用任何额外特性。或设置为以下选项中的一个或多个：

`All`

除 `MultiViews` 之外的所有特性。这是默认设置。

`ExecCGI`

允许使用 `mod_cgi` 执行CGI脚本。

`FollowSymLinks`

服务器允许在此目录中使用符号连接。

注意：即使服务器会使用符号连接，但它不会改变用于匹配 `<Directory>` 段的路径名。

注意：如果此配置位于 `<Location>` 配置段中，则此设置会被忽略。

`Includes`

允许使用 `mod_include` 提供的服务器端包含。

`IncludesNOEXEC`

允许服务器端包含，但禁用" `#exec cmd` "和" `#exec cgi` "。但仍可以从 `ScriptAlias` 目录使用" `#include virtual` "虚拟CGI脚本。

`Indexes`

如果一个映射到目录的URL被请求，而此目录中又没有 `DirectoryIndex` (例如： `index.html`)，那么服务器会返回由 `mod_autoindex` 生成的一个格式化后的目录列表。

`MultiViews`

允许使用 `mod_negotiation` 提供内容协商的"多重视图"(MultiViews)。

`SymLinksIfOwnerMatch`

服务器仅在符号连接与其目的目录或文件的拥有者具有相同的uid时才使用它。

注意

如果此配置出现在 `<Location>` 配置段中，此选项将被忽略。

一般来说，如果一个目录被多次设置了 `Options` ，则最特殊的一个会被完全接受(其它的被忽略)，而各个可选项的设定彼此并不融合(参见配置段的合并)。然而，如果所有作用于 `Options` 指令的可选项前都加有" `+` "或" `-` "符号，此可选项将被合并。所有前面加有" `+` "号的可选项将强制覆盖当前的可选项设置，而所有前面有" `-` "号的可选项将强制从当前可选项设置中去除。

比如，没有任何" `+` "或" `-` "符号：

```
     <Directory /web/docs>
Options Indexes FollowSymLinks
     </Directory>
     <Directory /web/docs/spec>
Options Includes
     </Directory>
```

那么只有将 `Includes` 设置到 `/web/docs/spec` 目录上。然而如果第二个 `Options` 指令使用了" `+` "和" `-` "符号：

```
     <Directory /web/docs>
Options Indexes FollowSymLinks
     </Directory>
     <Directory /web/docs/spec>
Options +Includes -Indexes
     </Directory>
```

那么就会有 `FollowSymLinks` 和 `Includes` 设置到 `/web/docs/spec` 目录上。

注意

使用 `-IncludesNOEXEC` 或 `-Includes` 时，不论前面如何设置，都会完全禁用服务器端包含。

没有其它设置时，默认设置为 `All` 。

Require 指令

说明	指定哪些认证用户允许访问该资源
语法	`Require entity-name [entity-name] ...`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	核心(C)
模块	core

这个指令指定哪些认证用户允许访问该资源。这些限制由授权支持模块实现。语法如下：

`Require user userid [userid] ...`

只有指定的用户可以访问此目录。

`Require group group-name [group-name] ...`

只有隶属于指定组的用户可以访问此目录。

`Require valid-user`

所有有效用户都可以访问此目录。

提供 `Require` 指令的授权支持模块有： `mod_authz_user` ， `mod_authz_groupfile` ， `mod_authnz_ldap` ， `mod_authz_dbm` ， `mod_authz_owner` 。

`Require` 必须伴随 `AuthName` 和 `AuthType` 指令，以及诸如 `AuthUserFile` 和 `AuthGroupFile` 指令(用以定义用户和用户组)以确保其能够正确工作。例如：

```
AuthType Basic
AuthName "Restricted Resource"
AuthUserFile /web/users
AuthGroupFile /web/groups
Require group admin
```

使用这种方法提供的访问控制对所有方法都有效。这是一般情况下期望达到的效果。如果您仅希望对某个特定的方法加以限制，而不涉及其它方法时，您可以将 `Require` 语句放入 `<Limit>` 配置段中。

如果 `Require` 与 `Allow` 或 `Deny` 指令同时使用，那么这些指令之间的相互作用由 `Satisfy` 指令控制。

在子目录中删除访问控制

下面的例子展示了如何使用 `Satisfy` 指令在一个受保护的目录下的子目录中取消访问控制。使用这种方法必须十分小心，因为它取消了 `mod_authz_host` 实现的任何访问控制。

```
      <Directory /path/to/protected/>

Require user david
      </Directory>

      <Directory /path/to/protected/unprotected>

# 该目录下的所有认证和访问控制都被取消了

        Satisfy Any

        Allow from all
      </Directory>
```

参见

- 认证、授权、访问控制
- `Satisfy`
- `mod_authz_host`

RLimitCPU 指令

说明	限制**Apache**子进程派生的进程占用**CPU**的最大秒数
语法	`RLimitCPU seconds|max [seconds|max]`
默认值	未定义，使用操作系统默认值
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

使用一个或两个参数。第一个参数设置了所有进程的软资源限制，第二个参数设置了最大资源限制。两个参数均可设置为一个数值或是" `max` "以表示设置为操作系统允许的最大值。增大此资源限制最大值需要以 `root` 运行服务器或是在初始化启动语句中进行设置。

这个限制将作用于Apache子进程服务的请求所衍生出的进程，而不是Apache子进程本身。这个范围包括CGI脚本和SSI执行命令，但不包括所有从Apache父进程衍生出的进程。比如管道日志。

CPU资源限制表示为每进程占用的秒数。

参见

- `RLimitMEM`
- `RLimitNPROC`

RLimitMEM 指令

说明	限制由Apache子进程派生的进程占用的最大内存字节数
语法	`RLimitMEM bytes|max [bytes|max]`
默认值	未定义，使用操作系统默认值
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

使用一个或两个参数。第一个参数设置了所有进程的软资源限制，第二个参数设置了最大资源限制。两个参数均可设置为一个数值或是" `max` "以表示设置为操作系统允许的最大值。增大此资源限制最大值需要以 `root` 运行服务器或是在初始化启动语句中进行设置。

这个限制将作用于Apache子进程服务的请求所衍生出的进程，而不是Apache子进程本身。这个范围包括CGI脚本和SSI执行命令，但不包括所有从Apache父进程衍生出的进程。比如管道日志。

内存资源限制表示为每进程占用的字节数。

参见

- `RLimitCPU`
- `RLimitNPROC`

RLimitNPROC 指令

说明	限制由Apache子进程派生的进程所派生的进程数目
语法	`RLimitNPROC number|max [number|max]`
默认值	未定义，使用操作系统默认值
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

使用一个或两个参数。第一个参数设置了所有进程的软资源限制，第二个参数设置了最大资源限制。两个参数均可设置为一个数值或是" `max` "以表示设置为操作系统允许的最大值。增大此资源限制最大值需要以 `root` 运行服务器或是在初始化启动语句中进行设置。

这个限制将作用于Apache子进程服务的请求所衍生出的进程，而不是Apache子进程本身。这个范围包括CGI脚本和SSI执行命令，但不包括所有从Apache父进程衍生出的进程。比如管道日志。

进程限制控制了每个用户的进程数。

注意

如果CGI进程不是以web服务器的uid启动的，那么这个指令将限制服务器自己能够创建的进程数目。此种情况将在 `error_log` 中以" **`cannot fork`** "进行记录。

参见

- `RLimitMEM`
- `RLimitCPU`

Satisfy 指令

说明	主机级别的访问控制和用户认证之间的相互关系
语法	`Satisfy Any|All`
默认值	`Satisfy All`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	核心(C)
模块	core
兼容性	2.0.51及以后版本中受 `<Limit>` 和 `<LimitExcept>` 指令影响

同时使用 `Allow` 和 `Require` 时的访问策略。参数可以设置为 `All` 或 `Any` 。这个指令仅在某个特定区域的访问控制同时被用户名/密码和客户端主机地址进行限定的时候起作用。默认行为(`All`)采取客户端首先通过地址访问限制并且输入有效的用户名和密码的方式。使用可选项 `Any` 将使客户端在通过主机限制或是输入一个有效的用户名和密码两种方式之一得到访问权限。这样，就可以通过密码来限制一个区域的访问，但允许某些特定地址的客户端访问时不需要输入密码。

比如，如果您想让您局域网内的用户访问您的web网站时不受限制，但局域网外的用户需提供密码才能进行访问，您可以采取类似如下的配置：

```
Require valid-user
Allow from 192.168.1
Satisfy Any
```

从2.0.51版本开始， `Satisfy` 指令可以被限定于由 `<Limit>` 和 `<LimitExcept>` 配置段指定的特定的方法。

参见

- `Allow`
- `Require`

ScriptInterpreterSource 指令

说明	定位CGI脚本解释器
语法	`ScriptInterpreterSource Registry|Registry-Strict|Script`
默认值	`ScriptInterpreterSource Script`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core
兼容性	仅用于Win32； `Registry-Strict` 选项仅在 Apache 2.0 及以后的版本中可用

这个指令控制Apache如何找到运行CGI脚本的解释器。默认为 `Script` ， 使用在脚本中以" `#!` "行指定的解释器。在Win32上这一行通常看上去像：

```
#!C:/Perl/bin/perl.exe
```

或者，如果 `perl` 的位置已经在 `PATH` 中指定，则可以简单的写成：

```
#!perl
```

设置为 `ScriptInterpreterSource Registry` 将使用脚本文件扩展名(比如 `.pl`)作为搜索项对Windows注册表项 `HKEY_CLASSES_ROOT` 进行搜索。 这个命令由注册表子键 `Shell\ExecCGI\Command` 或 `Shell\Open\Command` 定义(如果前者不存在)。如果无法找到该注册表项，Apache将采用默认值： `Script` 。

安全

将 `ScriptInterpreterSource Registry` 和使用了 `ScriptAlias` 的目录一起使用时需要非常小心，因为Apache会执行这个目录下的所有文件。 `Registry` 设置可能会导致对不可执行文件的不期望的程序调用。例如，在大多数Windows上默认打开 `.htm` 文件的程序是IE，所以任何一个对脚本目录中 `.htm` 文件的请求将会在服务器后台打开一个IE。这是一个让你的服务器在数分钟内崩溃的好办法。

`Registry-Strict` 选项和 `Registry` 差不多，但是只使用 `Shell\ExecCGI\Command` 子键。 `ExecCGI` 键不是一个普通的键。它必须在注册表中手动配置，从而可以防止意外的程序调用。

ServerAdmin 指令

说明	服务器返回给客户端的错误信息中包含的管理员邮件地址
语法	`ServerAdmin email-address|URL`
作用域	server config, virtual host
状态	核心(C)
模块	core

`ServerAdmin` 设置了在所有返回给客户端的错误信息中包含的管理员邮件地址。如果 `httpd` 不能将提供的参数识别为URL，它就会假定它是一个email-address，并在超连接中用在 `mailto:` 后面。推荐使用一个Email地址，因为许多CGI脚本是这样认为的。如果你确实想使用URL，一定要保证指向一个你能够控制的服务器，否则用户将无法确保一定可以和你取得联系。

为这个目的专门设置一个邮箱是值得的，比如：

```
ServerAdmin www-admin@foo.example.com
```

因为用户一般不会注意到他们在讨论服务器的问题！

ServerAlias 指令

说明	匹配一个基于域名的虚拟主机的别名
语法	`ServerAlias hostname [hostname] ...`
作用域	virtual host
状态	核心(C)
模块	core

`ServerAlias` 指令设定主机的别名，用于基于域名的虚拟主机。

```
<VirtualHost *>
ServerName server.domain.com
ServerAlias server server2.domain.com server2
# ...
</VirtualHost>
```

参见

- Apache虚拟主机文档

ServerName 指令

说明	服务器用于辨识自己的主机名和端口号
语法	`ServerName fully-qualified-domain-name[:port]`
作用域	server config, virtual host
状态	核心(C)
模块	core
兼容性	在2.0版中，这个指令代替了1.3版的 `Port` 指令的功能

`ServerName` 指令设置了服务器用于辨识自己的主机名和端口号。这主要用于创建重定向URL。比如，一个放置web服务器的主机名为 `simple.example.com` ，但同时有一个DNS别名 `www.example.com` 。而您希望web服务器更显著一点，您可以使用如下的指令：

```
ServerName www.example.com:80
```

当没有指定 `ServerName` 时，服务器会尝试对IP地址进行反向查询来推断主机名。如果在 `ServerName` 中没有指定端口号，服务器会使用接受请求的那个端口。为了加强可靠性和可预测性，您应该使用 `ServerName` 显式的指定一个主机名和端口号。

如果使用的是基于域名的虚拟主机，在 `<VirtualHost>` 段中的 `ServerName` 将是为了匹配这个虚拟主机，在" `Host:` "请求头中必须出现的主机名。

参见 `UseCanonicalName` 和 `UseCanonicalPhysicalPort` 指令以获得关于自引用URL(比如使用 `mod_dir` 模块)是需要指定一个特定端口，还是使用客户端请求的端口号的更详细的信息。

参见

- 关于DNS和Apache
- Apache虚拟主机文档
- `UseCanonicalName`
- `UseCanonicalPhysicalPort`
- `NameVirtualHost`
- `ServerAlias`

ServerPath 指令

说明	为兼容性不好的浏览器访问基于域名的虚拟主机保留的URL路径名
语法	`ServerPath URL-path`
作用域	virtual host
状态	核心(C)
模块	core

`ServerPath` 指令为主机设置了保守的(legacy)URL路径名，用于和基于域名的虚拟主机配合使用。

参见

- Apache虚拟主机文档

ServerRoot 指令

说明	安装服务器的基础目录
语法	`ServerRoot directory-path`
默认值	`ServerRoot /usr/local/apache`
作用域	server config
状态	核心(C)
模块	core

`ServerRoot` 指令设置了服务器所在的目录。一般来说它将包含 `conf/` 和 `logs/` 子目录。其它配置文件的相对路径即基于此目录 (比如 `Include` 或 `LoadModule`)。

示例

```
ServerRoot /home/httpd
```

参见

- `httpd` 的 `-d` 选项
- 安全提示中关于如何正确设置 `ServerRoot` 权限的部分

ServerSignature 指令

说明	配置服务器生成页面的页脚
语法	`ServerSignature On|Off|EMail`
默认值	`ServerSignature Off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	核心(C)
模块	core

`ServerSignature` 指令允许您配置服务器端生成文档的页脚(错误信息、 `mod_proxy` 的ftp目录列表、 `mod_info` 的输出)。您启用这个页脚的原因主要在于处于一个代理服务器链中的时候，用户基本无法辨识出究竟是链中的哪个服务器真正产生了返回的错误信息。

默认的 `Off` 设置没有错误行(这样便与Apache1.2及更旧版本兼容)。采用 `On` 会简单的增加一行关于服务器版本和正在伺服的虚拟主机的 `ServerName` ， 而 `EMail` 设置会如文档中说明的那样额外创建一个指向 `ServerAdmin` 的"mailto:"部分。

对于2.0.44以后的版本，显示的详细服务器版本号将由 `ServerTokens` 指令控制。

参见

- `ServerTokens`

ServerTokens 指令

说明	配置"`Server:` "应答头
语法	`ServerTokens Major|Minor|Min[imal]|Prod[uctOnly]|OS|Full`
默认值	`ServerTokens Full`
作用域	server config
状态	核心(C)
模块	core

这个指令控制了服务器回应给客户端的"`Server:` "应答头是否包含关于服务器操作系统类型和编译进的模块描述信息。

`ServerTokens Prod[uctOnly]`

服务器会发送(比如)： `Server: Apache`

`ServerTokens Major`

服务器会发送(比如)： `Server: Apache/2`

`ServerTokens Minor`

服务器会发送(比如)： `Server: Apache/2.0`

`ServerTokens Min[imal]`

服务器会发送(比如)： `Server: Apache/2.0.41`

`ServerTokens OS`

服务器会发送(比如)： `Server: Apache/2.0.41 (Unix)`

`ServerTokens Full` (或未指定)

服务器会发送(比如)： `Server: Apache/2.0.41 (Unix) PHP/4.2.2 MyMod/1.2`

此设置将作用于整个服务器，而且不能用在虚拟主机的配置段中。

2.0.44版本以后，这个指令还控制着 `ServerSignature` 指令的显示内容。

参见

- `ServerSignature`

SetHandler 指令

说明	强制所有匹配的文件被一个指定的处理器处理
语法	`SetHandler handler-name|None`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core
兼容性	在Apache2.0中移入核心

当这个指令放入 `.htaccess` 或 `<Directory>` 或 `<Location>` 配置段中时，这个指令将强制所有匹配的文件通过handler-name指定的处理器处理。比如：如果想不管某个目录中的文件具有什么扩展名，都将它作为图像映射规则文件来解析，您可以将下例放入那个目录的 `.htaccess` 中：

```
SetHandler imap-file
```

再来一个例子：如果您想当 `http://servername/status` 被请求时，服务器显示一个状态报告，您可以将下面的语句放入 `httpd.conf` 里面：

```
<Location /status>
SetHandler server-status
</Location>
```

你可以通过使用 `None` 来改写一个早先定义的 `SetHandler` 指令。

参见

- `AddHandler`

SetInputFilter 指令

说明	设置处理客户端请求和POST输入时使用的过滤器
语法	`SetInputFilter filter[;filter...]`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core

`SetInputFilter` 指令为服务器接受并处理客户端请求和POST输入设置了过滤器。这是在其它地方(包括 `AddInputFilter` 指令)设置的过滤器以外附加的过滤器。

如果设置了多于一个过滤器，它们必须按照处理内容的顺序用分号(;)分隔。

参见

- 过滤器文档

SetOutputFilter 指令

说明	设置用于处理服务器输出应答的过滤器
语法	`SetOutputFilter filter[;filter...]`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	核心(C)
模块	core

`SetOutputFilter` 指令设置了用于在服务器应答发送到客户端之前使用的过滤器。这是在其它地方(包括 `AddOutputFilter` 指令)设置的过滤器以外附加的过滤器。

例如：下述配置将会把 `/www/data/` 目录下的所有文件作为SSI文件来处理。

```
     <Directory /www/data/>

SetOutputFilter INCLUDES
     </Directory>
```

如果设置了多于一个过滤器，它们必须按照处理内容的顺序用分号(;)分隔。

参见

- 过滤器文档

TimeOut 指令

说明	服务器在断定请求失败前等待的秒数
语法	`TimeOut seconds`
默认值	`TimeOut 300`
作用域	server config
状态	核心(C)
模块	core

`TimeOut` 指令用于设置Apache等待以下三种事件的时间长度：

1. 接受一个GET请求耗费的总时间。
2. POST或PUT请求时，接受两个TCP包之间的时间。
3. 应答时TCP包传输中两个ACK包之间的时间。

我们计划在发展里程中，逐步把它们分别变得更易配置。计时器在1.2版本之前的默认值为1200，而现在已经设置为300了，但对于绝大多数情况来说仍是足够的。没有把它默认值设的更小的原因在于代码里还有点问题：有时发送一个包之后，计时器没有复位。

TraceEnable 指令

说明	确定如何处理 `TRACE` 请求
语法	`TraceEnable [on|off|extended]`
默认值	`TraceEnable on`
作用域	server config
状态	核心(C)
模块	core
兼容性	仅在 Apache 1.3.34, 2.0.55 及以后的版本中可用

这个指令同时决定了核心和 `mod_proxy` 模块如何处理 `TRACE` 请求。默认(`TraceEnable on`)允许处理RFC2616定义的禁止附带任何请求体的 `TRACE` 请求。 `TraceEnable off` 则使核心和 `mod_proxy` 模块向客户端返回一个" `405` "(禁止的方法)错误。

最后，为了测试和诊断，可以设置" `TraceEnable extended` "以允许附带请求体。核心(原始服务器)将会将请求体大小限制为64k以下(如果使用了 `Transfer-Encoding: chunked` 头，可以再为HTTP头增加8k)。核心会同时限制应答头和应答体；代理服务器则没有对请求体加以64k的限制。

UseCanonicalName 指令

说明	配置服务器如何确定它自己的域名和端口
语法	`UseCanonicalName On|Off|DNS`
默认值	`UseCanonicalName Off`
作用域	server config, virtual host, directory
状态	核心(C)
模块	core

在很多情况下，Apache必须构造一个_自引用_URL(即一个指回相同服务器的URL)。使用 `UseCanonicalName On` 会将 `ServerName` 这个域名用于所有自引用URL、 `SERVER_NAME` 、CGI中的 `SERVER_PORT` 。

设置为 `UseCanonicalName Off` 时，如果客户端提供了主机名和端口(否则将如上所述使用标准域名)，Apache将会使用这些信息来构建自引用URL。这些值与用于实现基于域名的虚拟主机的值相同，并且对于同样的客户端可用。CGI变量 `SERVER_NAME` 和 `SERVER_PORT` 也会由客户端提供的值来构建。

用这样的方式成功解决问题的例子如下：在一个局域网内，您想让连接主机的用户使用诸如 `www` 这样的短域名进行访问。您会注意到，如果用户键入了类似 `http://www/splat` 这样的短域名和目录的URL，并且没有最后的斜线，Apache会将他们重定向到 `http://www.domain.com/splat/` 。如果您在这个目录上启用了身份验证，这会导致用户必须进行两次验证：一次针对 `www` 而另外一次针对 `www.domain.com` (参见关于此主题的FAQ获得更多信息)。但如果 `UseCanonicalName` 设为 `Off` ，Apache就会重定向到 `http://www/splat/` 。

第三个可选项是 `UseCanonicalName DNS` ，用于为大量基于IP的虚拟主机支持那些古董级的不提供" `Host:` "头的浏览器使用。使用这个选项时，Apache将对客户端连入的服务器的IP地址进行反向DNS查询，以构建自引用URL。

警告

如果CGI对 `SERVER_NAME` 的值作出了假定，它们会被此选项破坏。客户端对于给出它们想要的主机名这样的行动是完全不受限制的。但如果CGI仅使用 `SERVER_NAME` 来构建自引用的URL，它们会工作的很好。

参见

- `UseCanonicalPhysicalPort`
- `ServerName`
- `Listen`

UseCanonicalPhysicalPort 指令

说明	配置服务器如何确定自己的名字和端口
语法	`UseCanonicalPhysicalPort On|Off`
默认值	`UseCanonicalPhysicalPort Off`
作用域	server config, virtual host, directory
状态	核心(C)
模块	core
兼容性	仅在 Apache 2.2.0 及以后的版本中可用

在很多情况下，Apache必须构造一个_自引用_URL(即一个指回相同服务器的URL)。在 `UseCanonicalPhysicalPort On` 的时候，Apache将有可能在构造服务器的规范端口时为了符合 `UseCanonicalName` 指令而使用实际的物理端口号(physical port)。在 `UseCanonicalPhysicalPort Off` 的时候，Apache将不会使用实际的物理端口号，而是依赖所有已经配置的信息来构造一个合法的端口号。

注意

决定使用物理端口号的次序如下：

`UseCanonicalName On`

- `Servername` 指定的端口号
- 物理端口号
- 默认端口号

`UseCanonicalName Off | DNS`

- " `Host:` "请求头提供的端口号
- 物理端口号
- `Servername` 指定的端口号
- 默认端口号

在 `UseCanonicalPhysicalPort Off` 的情况下，物理端口号将会从上述顺序中剔除。

参见

- UseCanonicalName
- ServerName
- Listen

<VirtualHost> 指令

说明	包含仅作用于指定主机名或IP地址的指令
语法	<VirtualHost addr[:port] [addr[:port]] ...> ... </VirtualHost>
作用域	server config
状态	核心(C)
模块	core

<VirtualHost> 和 </VirtualHost> 用于封装一组仅作用于特定虚拟主机的指令。任何在虚拟主机配置中可以使用的指令也同样可以在这里使用。当服务器接受了一个特定虚拟主机的文档请求时，它会使用封装在 <VirtualHost> 配置段中的指令。Addr可以是：

- 虚拟主机的IP地址
- 虚拟主机IP地址对应的完整域名
- 字符" * "，仅与" NameVirtualHost * "配合使用以匹配所有的IP地址
- 字符串" _default_ "，与基于IP的虚拟主机联用以捕获所有没有匹配的IP地址

示例

```
        <VirtualHost 10.1.2.3>
   ServerAdmin webmaster@host.foo.com
          DocumentRoot /www/docs/host.foo.com
          ServerName host.foo.com
          ErrorLog logs/host.foo.com-error_log
          TransferLog logs/host.foo.com-access_log
        </VirtualHost>
```

IPv6的地址必须放入方括号中指定，否则作为可选项的端口号将无法确定。一个IPv6的示例如下：

```
<VirtualHost [2001:db8::a00:20ff:fea7:ccea]>
ServerAdmin webmaster@host.example.com
DocumentRoot /www/docs/host.example.com
ServerName host.example.com
ErrorLog logs/host.example.com-error_log
TransferLog logs/host.example.com-access_log
</VirtualHost>
```

每个虚拟主机必须对应不同的IP地址、端口号或是不同的主机名。在第一种情况下，服务器所在的物理机器必须配置为可以为多个IP地址接受IP包。(在机器没有多个网络硬件界面的情况下，如果您的操作系统支持，您可以使用 `ifconfig alias` 命令来达到这个目的)。

注意

`<VirtualHost>` 的使用并不影响Apache的监听地址。你需要使用 `Listen` 来确保Apache正在监听正确的地址。

当使用基于IP的虚拟主机时，特殊的名称" `_default_` "可以在没有匹配到其它列出的虚拟主机的情况下作为匹配任何IP地址的默认虚拟主机。在没有进行" `_default_` "虚拟主机的设定时，在没有IP与请求匹配的情况下，将使用"主服务器"(在所有虚拟主机配置段之外)的配置。但请注意：任何匹配 `NameVirtualHost` 指令的IP地址既不会使用"主服务器"配置，也不会使用" `_default_` "虚拟主机的配置。参见基于域名的虚拟主机文档。

您可以指定一个" `:port` "来改变匹配的端口。如果没有指定，它将沿用主服务器中离它最近的那个 `Listen` 指定的值。您也可以指定" `:*` "来匹配那个地址上的所有端口(当您使用" `_default_` "时，这是推荐采用的方法)。

安全

参见安全提示文档以获得为什么当您存储日志文件的目录对于启动服务器以外的用户来说是可写的会危及服务器安全的详细资料。

参见

- Apache虚拟主机文档
- 关于DNS和Apache
- 设置Apache使用的地址和端口
- <Directory>、<Location>、<Files>配置段是如何工作的中包含了当接受一个请求时，这些不同的配置段是如何组合工作的相关解释。

Apache MPM 公共指令

说明	收集了被多个多路处理模块(MPM)实现的公共指令
状态	MPM

AcceptMutex 指令

说明	Apache用于串行化多个子进程在(多个)网络套接字(socket)上接受请求的方法
语法	`AcceptMutex Default|method`
默认值	`AcceptMutex Default`
作用域	server config
状态	MPM
模块	`prefork` , `worker`

`AcceptMutex` 指令用于设置串行化多个子进程在(多个)网络套接字上接受请求的方法。在2.0版本以前，只能在编译时设定此方法。应当在这里使用的最佳方法取决于不同的硬件体系结构和操作系统。欲知详情，请参见性能调节文档。

如果设置为 `Default` ，那么将会使用编译时自动选择的默认值。其他可用的方法在下面列出。注意，并不是所有的方法在所有的平台上都是可用的，如果指定了一个不可用的方法，将会在错误日志中记录下这个不可用的方法。

`flock`

这种方法调用系统函数 `flock(2)` 来锁定一个加锁文件(其位置取决于 `LockFile` 指令)。

`fcntl`

这种方法调用系统函数 `fcntl(2)` 来锁定一个加锁文件(其位置取决于 `LockFile` 指令)。

`posixsem`

(2.0及更新版本)这种方法使用了POSIX信号灯。如果一个运行中的线程占有了互斥segfault，则信号灯的所有者将不会被恢复，从而导致服务器的挂起和失去响应。

`pthread`

(1.3及更新版本)这种方法使用了POSIX互斥，按理应该可以用于所有完整实现了POSIX线程规范的体系中，但是似乎只能用在Solaris2.5及更新版本中，甚至只能在某种配置下才正常运作。如果遇到这种情况，则应该提防服务器的挂起和失去响应。只提供静态内容的服务器可能不受影响。

`sysvsem`

(1.3及更新版本)这种方案使用SysV风格的信号灯以实现互斥。不幸的是，SysV风格的信号灯有一些副作用，其一是，Apache有可能不能在结束以前释放这种信号灯(见 `ipcs()` 的man page)，另外，这种信号灯API给与网络服务器有相同uid的CGI提供了拒绝服务攻击的机会(所有CGI，除非用了类似 `suexec` 或 `cgiwrapper`)。鉴于此，在多数体系中都不用这种方法，除了IRIX(因为加锁文件的方法在IRIX中代价太高)。

如果你想知道编译时自动选择的默认值，你可以将 `LogLevel` 设为 `debug` ，这样默认的 `AcceptMutex` 就会记录到 `ErrorLog` 中。

警告

在大多数系统上，使用 `pthread` 时，如果一个子进程在持有 `AcceptCntl` 互斥信号时异常中止，服务器将会挂起和失去响应，此时必须手动重启服务器才能解决问题。但Solaris是一个例外，因为它提供了一个机制(Apache利用了该机制)，允许在一个持有互斥信号的子进程异常中止后恢复互斥信号。

如果你的操作系统实现了 `pthread_mutexattr_setrobust_np()` 函数，基本上就能安全的使用 `pthread` 选项。

CoreDumpDirectory 指令

说明	Apache使用的内核转储目录
语法	`CoreDumpDirectory directory`
默认值	参见说明
作用域	server config
状态	MPM
模块	`beos` , `mpm_winnt` , `prefork` , `worker`

这个指令用于控制Apache使用的内核转储目录。默认位于 `ServerRoot` 下，因为这个目录通常对于运行服务器的用户是不可写的，内核转储一般也就不会写入内容。如果你在调试中需要内核转储，你可以用这个指令来指定另外一个目录。

Linux上的内核转储

如果Apache以root身份启动并切换至其他用户，即使指定的转储目录对进程是可写的，Linux内核也将_禁止_Apache进行内核转储。但是Apache2.0.46及以后的版本在你明确指定 `CoreDumpDirectory` 的情况下，能够在Linux2.4以上的版本中强制实现内核转储。

EnableExceptionHook 指令

说明	在子进程崩溃以后启用一个钩子来运行异常处理程序
语法	`EnableExceptionHook On|Off`
默认值	`EnableExceptionHook Off`
作用域	server config
状态	MPM
模块	`prefork` , `worker`
兼容性	仅在 Apache 2.0.49 及以后的版本中可用

因为安全原因，这个指令仅在编译时使用了 `--enable-exception-hook` 选项的情况下才可用。它会在一个子进程崩溃以后启用一个钩子(hook)来运行一个外部模块以做些后继处理。

目前有两个模块(`mod_whatkilledus` 和 `mod_backtrace`)可以被钩子使用。请参见Jeff Trawick的EnableExceptionHook site以获得更多信息。

GracefulShutdownTimeout 指令

说明	指定优雅停止服务器的超时秒数
语法	`GracefulShutDownTimeout seconds`
默认值	`GracefulShutDownTimeout 0`
作用域	server config
状态	MPM
模块	`prefork` , `worker` , `event`
兼容性	仅在 Apache 2.2 及以后的版本中可用

`GracefulShutdownTimeout` 设置服务器在收到"优雅停止"信号后最多允许使用多少秒来处理尚未完成的连接，超时后服务器将强行退出。

设为"0"表示永不超时，也就是服务器必须在处理完所有尚未完成的请求之后才能退出。

Group 指令

说明	对请求提供服务的Apache子进程运行时的用户组
语法	`Group unix-group`
默认值	`Group #-1`
作用域	server config
状态	MPM
模块	`beos` , `mpmt_os2` , `prefork` , `worker`
兼容性	Apache2.0以后只对全局配置有效

`Group` 指令指定了用于对客户端请求提供服务的Apache子进程运行时的用户组。为了使用这个指令，Apache必须以 `root` 初始化启动，否则在切换用户组时会失败，并继续以初始化启动时的用户组运行。Unix-group可以是下列之一：

用户组的名称

通过名称引用组。

" `#` "号后跟一个组编号(GID)

通过编号引用组。

示例

```
Group www-group
```

建议你专门为Apache服务器新建一个用户组。一些管理员使用 `nobody` 用户，但是这并非总是可用或是合适的。

安全

不要将 `Group` (或 `User`)设置成 `root` ，除非你明确知道自己在做什么，并且明白其风险所在。

特别提示：在 `<VirtualHost>` 段中使用该指令已经不再被支持了。你可以使用 `suexec` 的 `SuexecUserGroup` 指令来达到这个目的。

注意

虽然 `Group` 指令也存在于 `beos` 和 `mpmt_os2` MPM中，但是事实上没用任何用处，只不过是个摆饰罢了。

Listen 指令

说明	服务器监听的IP地址和端口
语法	`Listen [IP-address:]portnumber [protocol]`
作用域	server config
状态	MPM
模块	`beos` , `mpm_netware` , `mpm_winnt` , `mpmt_os2` , `prefork` , `worker` , `event`
兼容性	Apache2.0以后必须设置该指令，protocol参数仅在2.1.5及以后版本中可用

`Listen` 指令指示Apache只在指定的IP地址和端口上监听；默认情况下Apache会在所有IP地址上监听。 `Listen` 是一个必须设置的指令。如果在配置文件中找不到这个指令，服务器将无法启动。这和先前的版本不一样。

`Listen` 指令指定服务器在那个端口或地址和端口的组合上监听接入请求。如果只指定一个端口，服务器将在所有地址上监听该端口。如果指定了地址和端口的组合，服务器将在指定地址的指定端口上监听。

使用多个 `Listen` 指令可以指定多个不同的监听端口和/或地址端口组合。服务器将会对列出的所有端口和地址端口组合上的请求作出应答。

例如，想要服务器接受80和8000端口上的请求，可以这样设置：

```
Listen 80
Listen 8000
```

为了让服务器在两个确定的地址端口组合上接受请求，可以这样设置：

```
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

IPv6地址必须像下面的例子一样，用方括号括起来：

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

可选的protocol参数在大多数情况下并不需要。若未指定该参数，则将为443端口使用默认的 `https` 协议，为其它端口使用 `http` 协议。在这里指定协议是为了确定使用哪个模块来处理请求，以及根据 `AcceptFilter` 指令根据不同的协议有针对性的进行优化。

仅在使用非标准端口时才需要指定protocol参数。比如在8443端口运行 `https` 协议：

```
Listen 192.170.2.1:8443 https
```

错误条件

多个 `Listen` 指令指定了同一个地址和端口的组合后，会导致" `Address already in use` "错误。

参见

- DNS问题
- 地址和端口绑定

ListenBackLog 指令

说明	半链接(pending connection)队列的最大长度
语法	`ListenBacklog backlog`
默认值	`ListenBacklog 511`
作用域	server config
状态	MPM
模块	`beos` , `mpm_netware` , `mpm_winnt` , `mpmt_os2` , `prefork` , `worker`

半链接(pending connection)队列的最大长度。一般不需要调整此项参数，然而在一些系统上，必须增大此值以抵御TCP SYN 洪水攻击。参见操作系统的 `listen(2)` 系统调用的后备参数。

操作系统常常将此值限制为一个较小的数字，具体根据操作系统的不同而不同。需要注意的是，许多操作系统并不是正好使用后备数值，而是取决于设置的值(通常大于后备值)。

LockFile 指令

说明	接受串行锁文件的位置
语法	`LockFile filename`
默认值	`LockFile logs/accept.lock`
作用域	server config
状态	MPM
模块	`prefork` , `worker`

`LockFile` 指令设置当 `AcceptMutex` 指令的值是 `fcntl` 或 `flock` 的时候，Apache使用的锁文件的位置。该指令通常保持它的默认值。改变默认值的主要原因是 `logs` 目录位于一个NFS文件系统上，因为锁文件必须位于本地磁盘上。主服务器进程的PID会自动添加到文件名后面。

安全

最好不要将此文件放在任何人都可以具有写权限的目录(比如 `/var/tmp`)中，因为别人可以通过建立一个与服务器企图建立的锁文件同名的文件，来阻止服务器启动，从而造成一个拒绝服务攻击。

参见

- `AcceptMutex`

MaxClients 指令

说明	允许同时伺服的最大接入请求数量
语法	`MaxClients number`
默认值	参见下面的说明
---	---
作用域	server config
状态	MPM
模块	`beos` , `prefork` , `worker`

`MaxClients` 指令设置了允许同时伺服的最大接入请求数量。任何超过 `MaxClients` 限制的请求都将进入等候队列，直到达到 `ListenBacklog` 指令限制的最大值为止。一旦一个链接被释放，队列中的请求将得到服务。

对于非线程型的MPM(也就是 `prefork`)， `MaxClients` 表示可以用于伺服客户端请求的最大子进程数量，默认值是 `256` 。要增大这个值，你必须同时增大 `ServerLimit` 。

对于线程型或者混合型的MPM(也就是 `beos` 或 `worker`)， `MaxClients` 表示可以用于伺服客户端请求的最大线程数量。线程型的 `beos` 的默认值是 `50` 。对于混合型的MPM默认值是 `16` (`ServerLimit`)乘以 `25` (`ThreadsPerChild`)的结果。因此要将 `MaxClients` 增加到超过16个进程才能提供的时候，你必须同时增加 `ServerLimit` 的值。

MaxMemFree 指令

说明	主内存分配程序在未调用 `free()` 的情况下允许持有的最大自由内存数量(KB)
语法	`MaxMemFree KBytes`
默认值	`MaxMemFree 0`
作用域	server config
状态	MPM
模块	`beos` , `mpm_netware` , `prefork` , `worker` , `mpm_winnt`

`MaxMemFree` 指令用于设置主内存分配程序在未调用 `free()` 的情况下允许持有的最大自由内存数量(KB)。若未设置或设置为"0"，将表示无限制。

MaxRequestsPerChild 指令

说明	每个子进程在其生存期内允许伺服的最大请求数量
语法	`MaxRequestsPerChild number`
默认值	`MaxRequestsPerChild 10000`
作用域	server config
状态	MPM
模块	`mpm_netware` , `mpm_winnt` , `mpmt_os2` , `prefork` , `worker`

`MaxRequestsPerChild` 指令设置每个子进程在其生存期内允许伺服的最大请求数量。到达 `MaxRequestsPerChild` 的限制后，子进程将会结束。如果 `MaxRequestsPerChild` 为" `0` "，子进程将永远不会结束。

不同的默认值

在 `mpm_netware` 和 `mpm_winnt` 上的默认值是" `0` "。

将 `MaxRequestsPerChild` 设置成非零值有两个好处：

- 可以防止(偶然的)内存泄漏无限进行，从而耗尽内存。
- 给进程一个有限寿命，从而有助于当服务器负载减轻的时候减少活动进程的数量。

注意

对于 `KeepAlive` 链接，只有第一个请求会被计数。事实上，它改变了每个子进程限制最大链接数量的行为。

MaxSpareThreads 指令

说明	最大空闲线程数
语法	`MaxSpareThreads number`
默认值	参见下面的说明
作用域	server config
状态	MPM
模块	`beos` , `mpm_netware` , `mpmt_os2` , `worker`

设置最大空闲线程数。不同的MPM对这个指令的处理是不一样的：

`worker` 的默认值是" `250` "。这个MPM将基于整个服务器监视空闲线程数。如果服务器中总的空闲线程数太多，子进程将杀死多余的空闲线程。

`mpm_netware` 的默认值是" `100` "。既然这个MPM只运行单独一个子进程，此MPM当然亦基于整个服务器监视空闲线程数。

`beos` 和 `mpmt_os2` 的工作方式与 `mpm_netware` 差不多， `beos` 的默认值是" `50` "； `mpmt_os2` 的默认值是" `10` "。

限制

`MaxSpareThreads` 的取值范围是有限制的。Apache将按照如下限制自动修正你设置的值：

- `mpm_netware` 要求其小于等于 `MinSpareThreads`
- `worker` 要求其大于等于 `MinSpareThreads` 加上 `ThreadsPerChild` 的和

参见

- `MinSpareThreads`
- `StartServers`

MinSpareThreads 指令

说明	最小空闲线程数
语法	`MinSpareThreads number`
默认值	参见下面的说明
作用域	server config
状态	MPM
模块	`beos` , `mpm_netware` , `mpmt_os2` , `worker`

设置最小空闲线程数，用于处理可能到来的突发请求。不同的MPM对这个指令的处理是不一样的：

`worker` 的默认值是" `75` "。这个MPM将基于整个服务器监视空闲线程数。如果服务器中总的空闲线程数太少，子进程将产生新的空闲线程。

`mpm_netware` 的默认值是" `10` "。既然这个MPM只运行单独一个子进程，此MPM当然亦基于整个服务器监视空闲线程数。

`beos` 和 `mpmt_os2` 的工作方式与 `mpm_netware` 差不多， `beos` 的默认值是" `1` "； `mpmt_os2` 的默认值是" `5` "。

参见

- `MaxSpareThreads`
- `StartServers`

PidFile 指令

说明	服务器用于记录父进程(监控进程)PID的文件
语法	`PidFile filename`
默认值	`PidFile logs/httpd.pid`
作用域	server config
状态	MPM
模块	`beos` , `mpm_winnt` , `mpmt_os2` , `prefork` , `worker`

`PidFile` 指令设置服务器用于记录父进程(监控进程)PID的文件。如果指定的不是绝对路径，那么将视为基于 `ServerRoot` 的相对路径。

示例

```
PidFile /var/run/apache.pid
```

这个文件通常用来便于给服务器父进程发送一个信号，用于关闭或重启服务器，以重新打开 `ErrorLog` 和 `TransferLog` 文件、重新读取配置文件。这些可以通过发送一个"SIGHUP"(kill -1)信号到 `PidFile` 记录的进程PID。

`PidFile` 和其他日志文件一样要注意放置位置和安全问题。

注意

从Apache2开始，推荐使用 `apachectl` 脚本来启动或停止服务器。

ReceiveBufferSize 指令

说明	TCP接收缓冲区大小(字节)
语法	`ReceiveBufferSize bytes`
默认值	`ReceiveBufferSize 0`
作用域	server config
状态	MPM
模块	`beos` , `mpm_netware` , `mpm_winnt` , `mpmt_os2` , `prefork` , `worker`

这个指令设置服务器的TCP接收缓冲区的大小(字节)。提高这个值会导致两个后果：高速度和高潜伏时间(100ms左右)。

如果设置为" `0` "，将使用操作系统默认值。

ScoreBoardFile 指令

说明	存储子进程协调数据(coordination data)的文件
语法	`ScoreBoardFile file-path`
默认值	`ScoreBoardFile logs/apache_status`
作用域	server config
状态	MPM
模块	`beos` , `mpm_winnt` , `prefork` , `worker`

Apache使用记分板(scoreboard)在父进程和子进程之间进行通信。一些体系结构要求有一个文件来帮助通信。如果未指定这个文件，Apache会首先尝试在匿名共享内存中建立完整的记分板(scoreboard)，若失败，将继续尝试使用基于文件的共享存储器在磁盘上建立这个文件。若利用这个指令指定这个文件的位置，则Apache将总是在磁盘上建立这个文件。

示例

```
ScoreBoardFile /var/run/apache_status
```

基于文件的共享存储器对于使用直接访问记分板(scoreboard)的第三方程序是很有用的。

将 `ScoreBoardFile` 放置在RAM disk中会对速度提升有很大帮助。但是同其他日志文件一样也要注意放置位置和安全问题。

参见

- 停止和重启Apache

SendBufferSize 指令

说明	TCP发送缓冲区大小(字节)
语法	`SendBufferSize bytes`
默认值	`SendBufferSize 0`
作用域	server config
状态	MPM
模块	`beos` , `mpm_netware` , `mpm_winnt` , `mpmt_os2` , `prefork` , `worker`

这个指令设置服务器的TCP发送缓冲区的大小(字节)。提高这个值会导致两个后果：高速度和高潜伏时间(100ms左右)。

如果设置为" `0` "，将使用操作系统默认值。

ServerLimit 指令

说明	服务器允许配置的进程数上限
语法	`ServerLimit number`
默认值	参见下面的说明
作用域	server config
状态	MPM
模块	`prefork` , `worker`

对于 `prefork` MPM，这个指令设置了 `MaxClients` 最大允许配置的数值。对于 `worker` MPM，这个指令和 `ThreadLimit` 结合使用设置了 `MaxClients` 最大允许配置的数值。任何在重启期间对这个指令的改变都将被忽略，但对 `MaxClients` 的修改却会生效。

使用这个指令时要特别当心。如果将 `ServerLimit` 设置成一个高出实际需要许多的值，将会有过多的共享内存被分配。如果将 `ServerLimit` 和 `MaxClients` 设置成超过系统的处理能力，Apache可能无法启动，或者系统将变得不稳定。

对于 `prefork` MPM，只有在你需要将 `MaxClients` 设置成高于默认值256的时候才需要使用这个指令。要将此指令的值保持和 `MaxClients` 一样。

对于 `worker` MPM，只有在你需要将 `MaxClients` 和 `ThreadsPerChild` 设置成需要超过默认值16个子进程的时候才需要使用这个指令。不要将该指令的值设置的比 `MaxClients` 需要的子进程数量高。

注意

Apache在编译时内部有一个硬限制" `ServerLimit 20000` "(对于 `prefork` MPM 为" `ServerLimit 200000` ")。你不能超越这个限制。

参见

- 停止和重启Apache

StartServers 指令

说明	服务器启动时建立的子进程数
语法	`StartServers number`
默认值	参见下面的说明
作用域	server config
状态	MPM
模块	`mpmt_os2` , `prefork` , `worker`

`StartServers` 指令设置了服务器启动时建立的子进程数量。因为子进程数量动态的取决于负载的轻重，所有一般没有必要调整这个参数。

不同的MPM默认值也不一样。对于 `worker` 默认值是" `3` "。对于 `prefork` 默认值是" `5` "， `mpmt_os2` 是" `2` "。

StartThreads 指令

说明	服务器启动时建立的线程数
语法	`StartThreads number`
默认值	参见下面的说明
作用域	server config
状态	MPM
模块	`beos` , `mpm_netware`

设置了服务器启动时建立的线程数量。因为线程数量动态的取决于负载的轻重，所有一般没有必要调整这个参数。

对于 `mpm_netware` ，默认值是" `50` "，由于只有一个进程，因此所有的线程都将用于伺服请求。

对于 `beos` ，默认值是" `10` "，同样也是所有的线程都将用于伺服请求。

ThreadLimit 指令

说明	每个子进程可配置的线程数上限
语法	`ThreadLimit number`
默认值	参见下面的说明
作用域	server config
状态	MPM
模块	`mpm_winnt` , `worker`
兼容性	仅用于2.0.41及以后版本的 `mpm_winnt`

这个指令设置了每个子进程可配置的线程数 `ThreadsPerChild` 上限。任何在重启期间对这个指令的改变都将被忽略，但对 `ThreadsPerChild` 的修改却会生效。

使用这个指令时要特别当心。如果将 `ThreadLimit` 设置成一个高出 `ThreadsPerChild` 实际需要很多的值，将会有过多的共享内存被分配。如果将 `ThreadLimit` 和 `ThreadsPerChild` 设置成超过系统的处理能力，Apache可能无法启动，或者系统将变得不稳定。该指令的值应当和 `ThreadsPerChild` 可能达到的最大值保持一致。

对于 `mpm_winnt` ， `ThreadLimit` 的默认值是 `1920` ；对于其他MPM这个值是 `64` 。

注意

Apache在编译时内部有一个硬性的限制" `ThreadLimit 20000` "(对于 `mpm_winnt` 是" `ThreadLimit 15000` ")，你不能超越这个限制。

ThreadsPerChild 指令

说明	每个子进程建立的线程数
语法	`ThreadsPerChild number`
默认值	参见下面的说明
作用域	server config
状态	MPM
模块	`mpm_winnt` , `worker`

这个指令设置了每个子进程建立的线程数。子进程在启动时建立这些线程后就不再建立新的线程了。如果使用一个类似于 `mpm_winnt` 只有一个子进程的MPM，这个数值要足够大，以便可以处理可能的请求高峰。如果使用一个类似于 `worker` 有多个子进程的MPM，每个子进程所

拥有的所有线程的总数要足够大，以便可以处理可能的请求高峰。

对于 `mpm_winnt` ， `ThreadsPerChild` 的默认值是 `64` ；对于其他MPM是 `25` 。

ThreadStackSize 指令

说明	处理客户端连接的线程使用的栈尺寸(字节)
语法	`ThreadStackSize size`
默认值	`NetWare上为65536；其它平台上等于操作系统默认值`
作用域	server config
状态	MPM
模块	`mpm_netware` , `mpm_winnt` , `worker`
兼容性	仅在 Apache 2.1 及以后的版本中可用

`ThreadStackSize` 指令设置了处理客户端连接(包括调用模块以协助处理)的线程允许使用的最大栈尺寸(字节)。在大多数情况下，操作系统默认的栈尺寸很合理，但是在某些情况下，需要调整这个值：

- 在默认栈尺寸较小的平台上(比如HP-UX)，Apache可能会在使用一些需要较大栈尺寸的第三方模块时崩溃。这样的问题可以通过将 `ThreadStackSize` 设置为一个较大的值来解决。这种调整应当仅仅在第三方模块提供者明确要求的情况下才需要，或者是您通过诊断确定是由于栈空间太小而导致崩溃。
- 在某些平台上，如果默认的栈空间大于服务器运行所需空间，那么将 `ThreadStackSize` 值降低到小于操作系统默认值可以让每个进程中允许生成的最大线程数量增加。这种类型的调整应该仅在测试环境中使用，并且对所有服务器进程进行充分的测试，因为处理某些罕见的请求需要较大的栈空间。一个很小的服务器配置变化就有可能使得当前的 `ThreadStackSize` 设置变得不合适。

User 指令

说明	实际服务于请求的子进程运行时的用户
语法	`User unix-userid`
默认值	`User #-1`
作用域	server config
状态	MPM
模块	`prefork` , `worker`
兼容性	2.0版本起仅在全局服务器配置中可用

`User` 指令用于设置实际提供服务的子进程的用户。为了使用这个指令，服务器必须以 `root` 身份启动和初始化。如果你以非 `root` 身份启动服务器，子进程将不能够切换至非特权用户，并继续以启动服务器的原始用户身份运行。如果确实以 `root` 用户启动了服务器，那么父进程将仍然以 `root` 身份运行。Unix-userid是下列值之一：

一个用户名

通过用户名引用用户

"#"号后面跟一个用户编号

通过用户编号引用用户

用于运行子进程的用户必须是一个没有特权的用户，这样才能保证子进程无权访问那些不想为外界所知的文件，同样的，该用户亦需没有执行那些不应当被外界执行的程序的权限。强烈推荐你专门为Apache子进程建立一个单独的用户和组。一些管理员使用 `nobody` 用户，但是这并不能总是符合要求，因为可能有其他程序也在使用这个用户。

安全

不要将 `User` (或 `Group`)设置成 `root` ，除非你明确知道自己在做什么，并且明白其风险所在。

特别提示：在 `<VirtualHost>` 段中使用该指令已经不再被支持了。你可以使用 `suexec` 的 `SuexecUserGroup` 指令来达到这个目的。

注意

虽然 `User` 指令也存在于 `beos` 和 `mpmt_os2` MPM中，但是事实上没用任何用处，只不过是个摆饰罢了。

Apache MPM beos

说明	This Multi-Processing Module is optimized for BeOS.
状态	MPM
模块名	mpm_beos_module
源文件	beos.c

概述

This Multi-Processing Module (MPM) is the default for BeOS. It uses a single control process which creates threads to handle requests.

MaxRequestsPerThread 指令

说明	Limit on the number of requests that an individual thread will handle during its life
语法	`MaxRequestsPerThread number`
默认值	`MaxRequestsPerThread 0`
作用域	server config
状态	MPM
模块	beos

`MaxRequestsPerThread` directive sets the limit on the number of requests that an individual server thread will handle. After `MaxRequestsPerThread` requests, the thread will die. If `MaxRequestsPerThread` is `0`, then the thread will never expire.

Setting `MaxRequestsPerThread` to a non-zero limit has two beneficial effects:

- it limits the amount of memory that a thread can consume by (accidental) memory leakage;
- by giving threads a finite lifetime, it helps reduce the number of threads when the server load reduces.

注意：

For `KeepAlive` requests, only the first request is counted towards this limit. In effect, it changes the behavior to limit the number of *connections* per thread.

Apache MPM event

说明	**An experimental variant of the standard `worker` MPM**
状态	MPM
模块名	mpm_event_module
源文件	event.c

概述

警告

This MPM is experimental, so it may or may not work as expected.

To use the `event` MPM, add `--with-mpm=event` to the `configure` script's arguments when building the `httpd` .

This MPM depends on APR's atomic compare-and-swap operations for thread synchronization. If you are compiling for an x86 target and you don't need to support 386s, or you are compiling for a SPARC and you don't need to run on pre-UltraSPARC chips, add `--enable-nonportable-atomics=yes` to the `configure` script's arguments. This will cause APR to implement atomic operations using efficient opcodes not available in older CPUs.

Apache MPM netware

说明	**Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare**
状态	MPM
模块名	mpm_netware_module
源文件	mpm_netware.c

概述

This Multi-Processing Module (MPM) implements an exclusively threaded web server that has been optimized for Novell NetWare.

The main thread is responsible for launching child worker threads which listen for connections and serve them when they arrive. Apache always tries to maintain several <dfn class="calibre27">spare</dfn> or idle worker threads, which stand ready to serve incoming requests. In this way, clients do not need to wait for a new child threads to be spawned before their requests can be served.

`StartThreads` , `MinSpareThreads` , `MaxSpareThreads` , and `MaxThreads` regulate how the main thread creates worker threads to serve requests. In general, Apache is very self-regulating, so most sites do not need to adjust these directives from their default values. Sites with limited memory may need to decrease `MaxThreads` to keep the server from thrashing (spawning and terminating idle threads). More information about tuning process creation is provided in the performance hints documentation.

`MaxRequestsPerChild` controls how frequently the server recycles processes by killing old ones and launching new ones. On the NetWare OS it is highly recommended that this directive remain set to 0. This allows worker threads to continue servicing requests indefinitely.

MaxThreads 指令

说明	Set the maximum number of worker threads
语法	`MaxThreads number`
默认值	`MaxThreads 2048`
作用域	server config
状态	MPM
模块	mpm_netware

`MaxThreads` directive sets the desired maximum number worker threads allowable. The default value is also the compiled in hard limit. Therefore it can only be lowered, for example:

```
MaxThreads 512
```

Apache MPM os2

说明	Hybrid multi-process, multi-threaded MPM for OS/2
状态	MPM
模块名	mpm_mpmt_os2_module
源文件	mpmt_os2.c

概述

The Server consists of a main, parent process and a small, static number of child processes.

The parent process's job is to manage the child processes. This involves spawning children as required to ensure there are always `StartServers` processes accepting connections.

Each child process consists of a a pool of worker threads and a main thread that accepts connections and passes them to the workers via a work queue. The worker thread pool is dynamic, managed by a maintenance thread so that the number of idle threads is kept between `MinSpareThreads` 和 `MaxSpareThreads` .

Apache MPM prefork

说明	一个非线程型的、预派生的**MPM**
状态	MPM
模块名	mpm_prefork_module
源文件	prefork.c

概述

这个多路处理模块(MPM)实现了一个非线程型的、预派生的web服务器，它的工作方式类似于Apache 1.3。它适合于没有线程安全库，需要避免线程兼容性问题的系统。它是要求将每个请求相互独立的情况下最好的MPM，这样若一个请求出现问题就不会影响到其他请求。

这个MPM具有很强的自我调节能力，只需要很少的配置指令调整。最重要的是将 `MaxClients` 设置为一个足够大的数值以处理潜在的请求高峰，同时又不能太大，以致需要使用的内存超出物理内存的大小。

工作方式

一个单独的控制进程(父进程)负责产生子进程，这些子进程用于监听请求并作出应答。Apache总是试图保持一些<dfn class="calibre27">备用的(spare)</dfn>或者是空闲的子进程用于迎接即将到来的请求。这样客户端就不需要在得到服务前等候子进程的产生。

`StartServers` , `MinSpareServers` , `MaxSpareServers` , `MaxClients` 指令用于调节父进程如何产生子进程。通常情况下Apache具有很强的自我调节能力，所以一般的网站不需要调整这些指令的默认值。可能需要处理最大超过256个并发请求的服务器可能需要增加 `MaxClients` 的值。内存比较小的机器则需要减少 `MaxClients` 的值以保证服务器不会崩溃。更多关于调整进程产生的问题请参见性能方面的提示。

在Unix系统中，父进程通常以 `root` 身份运行以便邦定80端口，而Apache产生的子进程通常以一个低特权的用户运行。 `User` 和 `Group` 指令用于设置子进程的低特权用户。运行子进程的用户必须要对它所服务的内容有读取的权限，但是对服务内容之外的其他资源必须拥有尽可能少的权限。

`MaxRequestsPerChild` 指令控制服务器杀死旧进程产生新进程的频率。

MaxSpareServers 指令

说明	空闲子进程的最大数量
语法	`MaxSpareServers number`
默认值	`MaxSpareServers 10`
作用域	server config
状态	MPM
模块	prefork

`MaxSpareServers` 指令设置空闲子进程的最大数量。所谓空闲子进程是指没有正在处理请求的子进程。如果当前有超过 `MaxSpareServers` 数量的空闲子进程，那么父进程将杀死多余的子进程。

只有在非常繁忙机器上才需要调整这个参数。将此参数设的太大通常是一个坏主意。如果你将该指令的值设置为比 `MinSpareServers` 小，Apache将会自动将其修改成" `MinSpareServers``+1` "。

参见

- `MinSpareServers`
- `StartServers`

MinSpareServers 指令

说明	空闲子进程的最小数量
语法	`MinSpareServers number`
默认值	`MinSpareServers 5`
作用域	server config
状态	MPM
模块	prefork

`MinSpareServers` 指令设置空闲子进程的最小数量。所谓空闲子进程是指没有正在处理请求的子进程。如果当前空闲子进程数少于 `MinSpareServers` ，那么Apache将以最大每秒一个的速度产生新的子进程。

只有在非常繁忙机器上才需要调整这个参数。将此参数设的太大通常是一个坏主意。

参见

- `MaxSpareServers`

- `StartServers`

Apache MPM winnt

说明	专门为**Windows NT**优化过的**MPM**
状态	MPM
模块名	mpm_winnt_module
源文件	mpm_winnt.c

概述

该多路处理模块(MPM)是Windows NT上的默认值。它使用一个单独的父进程产生一个单独的子进程，在这个子进程中轮流产生多个线程来处理请求。

Win32DisableAcceptEx 指令

说明	使用**accept()**代替**AcceptEx()**接受网络链接
语法	`Win32DisableAcceptEx`
默认值	`AcceptEx()` 是默认的，使用这个指令将禁用它。
作用域	server config
状态	MPM
模块	mpm_winnt
兼容性	仅在 Apache 2.0.49 及以后的版本中可用

`AcceptEx()` 是一个微软的WinSock2 API ，通过使用BSD风格的 `accept()` API提供了性能改善。一些流行的Windows产品，比如防病毒软件或虚拟专用网络软件，会干扰 `AcceptEx()` 的正确操作。如果你遇到类似于如下的错误：

```
[error] (730038)An operation was attempted on something that is
not a socket.: winnt_accept: AcceptEx failed. Attempting to recover.
```

你就需要使用这个指令来禁止使用 `AcceptEx()` 。

Apache MPM worker

说明	支持混合的多线程多进程的多路处理模块
状态	MPM
模块名	mpm_worker_module
源文件	worker.c

概述

此多路处理模块(MPM)使网络服务器支持混合的多线程多进程。由于使用线程来处理请求，所以可以处理海量请求，而系统资源的开销小于基于进程的MPM。但是，它也使用了多进程，每个进程又有多个线程，以获得基于进程的MPM的稳定性。

控制这个MPM的最重要的指令是，控制每个子进程允许建立的线程数的 `ThreadsPerChild` 指令，和控制允许建立的总线程数的 `MaxClients` 指令。

工作方式

每个进程可以拥有的线程数量是固定的。服务器会根据负载情况增加或减少进程数量。一个单独的控制进程(父进程)负责子进程的建立。每个子进程可以建立 `ThreadsPerChild` 数量的服务线程和一个监听线程，该监听线程监听接入请求并将其传递给服务线程处理和应答。

Apache总是试图维持一个<dfn class="calibre27">备用(spare)</dfn>或是空闲的服务线程池。这样，客户端无须等待新线程或新进程的建立即可得到处理。初始化时建立的进程数量由 `StartServers` 指令决定。随后父进程检测所有子进程中空闲线程的总数，并新建或结束子进程使空闲线程的总数维持在 `MinSpareThreads` 和 `MaxSpareThreads` 所指定的范围内。由于这个过程是自动调整的，几乎没有必要修改这些指令的缺省值。可以并行处理的客户端的最大数量取决于 `MaxClients` 指令。活动子进程的最大数量取决于 `MaxClients` 除以 `ThreadsPerChild` 的值。

有两个指令设置了活动子进程数量和每个子进程中线程数量的硬限制。要想改变这个硬限制必须完全停止服务器然后再启动服务器(直接重启是不行的)，`ServerLimit` 是活动子进程数量的硬限制，它必须大于或等于 `MaxClients` 除以 `ThreadsPerChild` 的值。`ThreadLimit` 是所有服务线程总数的硬限制，它必须大于或等于 `ThreadsPerChild` 指令。这两个指令必须出现在其他 `worker` MPM指令的前面。

在设置的活动子进程数量之外，还可能有额外的子进程处于"正在中止"的状态但是其中至少有一个服务线程仍然在处理客户端请求，直到到达 `MaxClients` 以致结束进程，虽然实际数量会很小。这个行为能够通过以下禁止特别的子进程中止的方法来避免：

- 将 `MaxRequestsPerChild` 设为"0"
- 将 `MaxSpareThreads` 和 `MaxClients` 设为相同的值

一个典型的针对 `worker` MPM的配置如下：

```
ServerLimit         16
StartServers         2
MaxClients         150
MinSpareThreads     25
MaxSpareThreads     75
ThreadsPerChild     25
```

在Unix中，为了能够绑定80端口，父进程一般都是以 `root` 身份启动，随后，Apache以较低权限的用户建立子进程和线程。 `User` 和 `Group` 指令用于设置Apache子进程的权限。虽然子进程必须对其提供的内容拥有读权限，但应该尽可能给予它较少的特权。另外，除非使用了 `suexec` ，否则，这些指令设置的权限将被CGI脚本所继承。

`MaxRequestsPerChild` 指令用于控制服务器建立新进程和结束旧进程的频率。

Apache模块 mod_actions

说明	基于媒体类型或请求方法，为执行CGI脚本而提供
状态	基本(B)
模块名	actions_module
源文件	mod_actions.c

概述

此模块有两个指令。 `Action` 指令让你可以在对特定MIME类型文件请求的时候运行CGI脚本。 `Script` 指令让你能够在使用特定请求方法的时候运行CGI脚本。这使得执行处理文件的CGI脚本更加容易。

Action 指令

说明	针对特定的处理器或内容类型激活一个CGI脚本
语法	`Action action-type cgi-script [virtual]`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_actions
兼容性	`virtual` 修饰词仅在 Apache 2.1 及之后的版本中可用

这条指令添加一个当action-type被请求触发时会执行cgi-script的动作。cgi-script是一个URL路径，指向一个已经被用 `ScriptAlias` 或 `AddHandler` 指令指定为CGI脚本的资源。action-type可以是一个处理器或一个MIME内容类型。它使用标准的 `PATH_INFO` 和 `PATH_TRANSLATED` 环境变量来发送此URL和被请求内容的文件路径。用于该请求的处理器通过 `REDIRECT_HANDLER` 变量传递。

例子

```
# 对特定MIME内容类型文件的请求
Action image/gif /cgi-bin/images.cgi
# 对于具有特定扩展名的文件
AddHandler my-file-type .xyz
Action my-file-type /cgi-bin/program.cgi
```

在第一个例子中，对于所有对MIME类型" `image/gif` "的请求都将被指定的CGI脚本 `/cgi-bin/images.cgi` 进行处理。

在第二个例子中，对于所有对具有扩展名" `.xyz` "的文件的请求都将被指定的CGI脚本 `/cgi-bin/program.cgi` 进行处理。

可选的 `virtual` 修饰词关闭了对所请求的文件是否真实存在的检查。这个修饰词很有用，比如希望将 `Action` 使用于虚拟位置的时候：

示例

```
<Location /news>
SetHandler news-handler
Action news-handler /cgi-bin/news.cgi virtual
</Location>
```

参见

- `AddHandler`

Script 指令

说明	对特定的请求方法激活一个CGI脚本
语法	`Script method cgi-script`
作用域	server config, virtual host, directory
状态	基本(B)
模块	mod_actions

该指令添加一个当文件被method方法请求时会激活脚本cgi-script的动作。cgi-script是一个URL路径，指向一个已用 `ScriptAlias` 或 `AddHandler` 指令指定为CGI脚本的资源。它使用标准的 `PATH_INFO` 和 `PATH_TRANSLATED` 环境变量来发送此URL和被请求内容的文件路径。

可以使用任意的方法名称。方法名大小写敏感，因此 `Script PUT` 和 `Script put` 具有完全不同的结果。

注意，`Script`命令只定义了默认的动作。如果一个CGI脚本或其他能够内部处理此请求的资源被调用，就将这样做。同时注意对应`GET`方法的脚本只有在提供了查询参数的时候才会被调用(例如："foo.html?hi")。否则，该请求将被正常处理。

例子

```
# <ISINDEX>风格的搜索

Script GET /cgi-bin/search

# A CGI PUT 处理器

Script PUT /~bob/put.cgi
```

Apache模块 mod_alias

说明	提供从文件系统的不同部分到文档树的映射和URL重定向
状态	基本(B)
模块名	alias_module
源文件	mod_alias.c

概述

此模块提供的指令可以操控作为请求到达服务器的URL。 `Alias` 和 `ScriptAlias` 指令用于在URL和文件系统路径之间实现映射，使不在 `DocumentRoot` 目录下的内容也能成为文档树的一部分，其中， `ScriptAlias` 指令有更多一层的含义，它标明此目标目录下只有CGI脚本。

`Redirect` 指令引导客户端以一个不同的URL产生一个新的请求，常用于一个资源被移动到一个新位置的时候。

`mod_alias` 被设计成处理普通的URL操作。复杂的URL操作，比如处理请求字符串，请使用 `mod_rewrite` 提供的强大功能。

处理顺序

出现在不同作用域(context)中的别名指令以及重定向指令和其他指令一样，按照标准的合并规则进行处理。但是当多个别名指令或重定向指令出现在同一个作用域(context)中的时候(比如在同一个 `<VirtualHost>` 段)，处理顺序就比较特别了：

首先，所有重定向指令都优先于别名指令被处理，因此一个匹配 `Redirect` 或 `RedirectMatch` 的请求将永远不会被别名指令处理。其次，别名指令和重定向指令将按照他们在配置文件中出现的先后顺序进行匹配，并由最先匹配到的指令进行处理。

因为这个原因，当两个或两个以上的这些指令作用于同一个子路径时，你必须将最特殊的路径放在最前面，以便所有指令都能正确地生效。例如下面的例子将按照你原本的意愿正常工作：

```
Alias /foo/bar /baz

Alias /foo /gaq
```

但是，如果将上面两条指令的顺序颠倒，则后一条指令永远也得不到匹配的机会。

Alias 指令

说明	映射**URL**到文件系统的特定区域
语法	`Alias URL-path file-path|directory-path`
作用域	server config, virtual host
状态	基本(B)
模块	mod_alias

`Alias` 指令使文档可以被存储在 `DocumentRoot` 以外的本地文件系统中。以(%已解码的)url-path路径开头的URL可以被映射到以directory-path开头的本地文件。

示例：

```
      Alias /image /ftp/pub/image
```

对"http://myserver/image/foo.gif"的请求，服务器将返回"/ftp/pub/image/foo.gif"文件。因为仅匹配完整路径，所以上述例子不会匹配对"http://myserver/imagefoo.gif"的请求。对于使用正则表达式的匹配，请参见`AliasMatch`指令。

注意：如果url-path中有后缀"/"，则服务器要求有后缀"/"以扩展此别名。也就是说" `Alias /icons/ /usr/local/apache/icons/` "并不能对" `/icons` "实现别名。

注意，可能需要额外指定一个 `<Directory>` 段来覆盖别名的最终对象。由于只有出现在 `<Directory>` 段之前的别名才会被检测，所以它只对最终对象生效。(由于执行别名操作之前 `<Location>` 段会被首先扫描一次，所以它们也是有效的)

特别地，如果对在 `DocumentRoot` 之外的某个目录建立了一个 `Alias` ，则可能需要明确的对目标目录设定访问权限。

示例：

```
        Alias /image /ftp/pub/image

        <Directory /ftp/pub/image>

  Order allow,deny

            Allow from all
        </Directory>
```

AliasMatch 指令

说明	使用正则表达式映射URL到文件系统
语法	`AliasMatch regex file-path|directory-path`
作用域	server config, virtual host
状态	基本(B)
模块	mod_alias

这个指令与 `Alias` 等效，但是它使用了标准的正则表达式，而不是简单的前缀匹配。如果此正则表达式与URL-path相匹配，则服务器会把所匹配的括弧中的字符串替换到该指令所指定的目标字符串中，并视之为一个文件名。例如，要使用" `/icons` "目录，可以：

```
AliasMatch ^/icons(.*) /usr/local/apache/icons$1
```

Redirect 指令

说明	发送一个外部重定向使客户端重定向到一个不同的URL
语法	`Redirect [status] URL-path URL`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_alias

该指令将一个老URL映射为一个新URL，此新URL被返回到客户端使之重定向到一个新地址。

老*URL-path*是一个(%已解码的)以"/"开头的(网络空间)绝对路径。新*URL*是一个(%已编码的)以"/"开头的(网络空间)绝对路径或者包含协议名和主机名的完整URL。当新*URL*不包含协议名和主机名时将使用与老*URL-path*相同的当前值。

这样，对任何以老*URL-path*开头的请求，将返回一个指向以新*URL*开头的重定向应答。

示例：

```
Redirect /service http://foo2.example.com/service
```

如果客户端请求"http://example.com/service/foo.txt"，则会被重定向到"http://foo2.example.com/service/foo.txt"。因为仅匹配完整路径，所以上述例子不会匹配"http://example.com/servicefoo.txt"请求。对于使用正则表达式的匹配，请参见`RedirectMatch`指令。

注意

重定向指令总是优先于Alias和ScriptAlias指令，而无论他们在配置文件中的顺序如何。

如果没有指定status参数，则重定向是"临时的"(HTTP status 302)。也就是对客户端来说，此资源的位置变动是临时性的。此status参数可以返回以下HTTP状态码：

permanent

返回一个永久性重定向状态码(301)，表示此资源的位置变动是永久性的。

temp

返回一个临时性重定向状态码(302)，这是默认值。

seeother

返回一个"参见"状态码(303)，表示此资源已经被替代。

gone

返回一个"已废弃"状态码(410)，表示此资源已经被永久性地删除了。如果指定了这个状态码，则URL参数将被忽略。

status可以被指定为数字状态以返回其他状态码。如果此状态在300-399之间，则必须提供URL参数，否则将被忽略。注意，此状态码必须是Apache已知的(参见http_protocol.c中的 `send_error_response` 函数)。

示例：

```
Redirect permanent /one http://example.com/two

Redirect 303 /three http://example.com/other
```

RedirectMatch 指令

说明	基于正则表达式匹配对当前的URL发送一个外部重定向
语法	`RedirectMatch [status] regex URL`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_alias

该指令与 `Redirect` 等效，但是它使用了标准的正则表达式，而不是简单的前缀匹配。如果regex与URL-path相匹配，则服务器会把所匹配的括弧中的字符串替换到该指令所指定的目标字符串中，并视之为一个文件名。例如，重定向所有GIF文件到另一个服务器上同名的JPEG文件，可以：

```
RedirectMatch (.*)\.gif$ http://www.anotherserver.com$1.jpg
```

RedirectPermanent 指令

说明	发送一个外部永久重定向使客户端重定向到一个不同的URL
语法	`RedirectPermanent URL-path URL`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_alias

此指令告知客户端此重定向是永久性的(status 301)。与" `Redirect permanent` "等效。

RedirectTemp 指令

说明	发送一个外部临时重定向使客户端重定向到一个不同的URL
语法	`RedirectTemp URL-path URL`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_alias

此指令告知客户端此重定向只是临时性的(status 302)。与" `Redirect temp` "等效。

ScriptAlias 指令

说明	映射一个**URL**到文件系统并视之为**CGI**脚本
语法	`ScriptAlias URL-path file-path|directory-path`
作用域	server config, virtual host
状态	基本(B)
模块	mod_alias

`ScriptAlias` 指令的行为与 `Alias` 指令相同，但同时它又标明此目录中含有应该由cgi-script处理器处理的CGI脚本。以URL-path开头的(%已解码的)的URL会被映射到由第二个参数指定的具有完整路径名的本地文件系统中的脚本。

示例：

```
ScriptAlias /cgi-bin/ /web/cgi-bin/
```

对 `http://myserver/cgi-bin/foo` 的请求会引导服务器执行 `/web/cgi-bin/foo` 脚本。

ScriptAliasMatch 指令

说明	使用正则表达式映射一个**URL**到文件系统并视之为**CGI**脚本
语法	`ScriptAliasMatch regex file-path|directory-path`
作用域	server config, virtual host
状态	基本(B)
模块	mod_alias

该指令与 `ScriptAlias` 等效，但是它使用了标准的正则表达式，而不是简单的前缀匹配。如果regex与URL-path相匹配，则服务器会把所匹配的括弧中的字符串替换到该指令所指定的目标字符串中，并视之为一个文件名。例如，要使用标准的 `/cgi-bin` ，可以：

```
ScriptAliasMatch ^/cgi-bin(.*) /usr/local/apache/cgi-bin$1
```

Apache模块 mod_asis

说明	发送自己包含HTTP头内容的文件
状态	基本(B)
模块名	asis_module
源文件	mod_asis.c

概述

这个模块提供了 `send-as-is` 处理器，这样Apache可以不加大多数常用的HTTP头(headers)传送它们。

这可以用来从服务器传送任何型态的资料，包括重定向以及其它特殊的HTTP应答，而不需要cgi-script或是nph script 。

由于历史原因，这个模块也处理MIME类型为 `httpd/send-as-is` 的文件。

用法

在服务器配置文档里，定义一个称为 `send-as-is` 的处理器，例如：

```
AddHandler send-as-is asis
```

任何带有" `.asis` "扩展名的文件的内容被Apache发往客户端时几乎没有什么变化。客户端将需要HTTP头来联系，所以别忘记它们。"Status:"是必须的头；此数据应该是3位数字的HTTP应答码，跟随一段文字信息。

这里有个示例文档，其内容将按照原样输出，它是告诉客户端重定向到另外一个文件：

```
Status: 301 Now where did I leave that URL

Location: http://xyz.abc.com/foo/bar.html

Content-type: text/html

<html>

<head>

<title>Lame excuses'R'us</title>

</head>

<body>

<h1>Fred's exceptionally wonderful page has moved to

<a href="http://xyz.abc.com/foo/bar.html">Joe's</a>
site.

</h1>

</body>

</html>
```

注意：

服务器总会在送给客户端的资料里加上"`Date:`"和"`Server:`"头，所以这两个头不应该包含在这个文件里。另外，服务器不会加上通常都会加上的"`Last-Modified:`"头。

Apache模块 mod_auth_basic

说明	使用基本认证
状态	基本(B)
模块名	auth_basic_module
源文件	mod_auth_basic.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

此模块使用HTTP基本认证，在认证支持模块的帮助下查找用户名和密码，从而进行访问控制。而 `mod_auth_digest` 模块则提供了对HTTP摘要认证的支持。这两个模块通常至少需要和一个认证支持模块(如 `mod_authn_file`)和一个授权支持模块(如 `mod_authz_user`)一起使用。

AuthBasicAuthoritative 指令

说明	指定是否将(基本)认证和授权操作交由更底层的模块来处理
语法	`AuthBasicAuthoritative On|Off`
默认值	`AuthBasicAuthoritative On`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	基本(B)
模块	mod_auth_basic

通常，在 `AuthBasicProvider` 中列出的每一个认证支持者都会尝试校验用户，如果所有认证支持者全都没有通过检验，那么访问将被拒绝。在将 `AuthBasicAuthoritative` 明确地设置为 `Off` 的情况下，如果提供的userID不能与任何**userID**或**rule(**规则**)**相匹配，则认证和授权操作均转交其它非认证支持(non-provider-based)模块(也就是第三方模块)来处理；仅仅在将 `mod_auth_basic` 和未使用 `AuthBasicProvider` 进行配置的第三方模块联合使用的时候才需要这样设置。在使用这种第三方模块的时候，处理顺序是在该模块的源代码中设定的，不能被重新配置。

AuthBasicProvider 指令

说明	设置该区域的(基本)认证支持者(Provider)
语法	`AuthBasicProvider provider-name [provider-name] ...`
默认值	`AuthBasicProvider file`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	基本(B)
模块	mod_auth_basic

`AuthBasicProvider` 指令设置了该区域的(基本)认证支持者(Provider)。默认的 `file` 支持者由 `mod_authn_file` 模块实现。必须确保所需的认证支持模块存在于服务器中(静态连接或DSO)。

示例

```
      <Location /secure>
AuthType basic
        AuthBasicProvider  dbm
        AuthDBMType        SDBM
        AuthDBMUserFile    /www/etc/dbmpasswd
        Require            valid-user
      </Location>
```

能够提供认证支持者(Provider)的模块如下：`mod_authn_dbm` , `mod_authn_file` , `mod_authn_dbd` , `mod_authnz_ldap` 。

Apache模块 mod_auth_digest

说明	使用MD5摘要认证(更安全，但是只有最新的浏览器才支持)
状态	实验(X)
模块名	auth_digest_module
源文件	mod_auth_digest.c

概述

这个模块实现了HTTP摘要认证。由于尚未进行过广泛的测试，因此标记为试验模块。

使用摘要认证

使用MD5摘要认证很简单。简单的安装认证模块后使用" `AuthType Digest` "和 `AuthDigestProvider` 代替普通的" `AuthType Basic` "和 `AuthBasicProvider` ，再添加一个 `AuthDigestDomain` 指令包含至少是需要保护的区域的根URI。

可以使用 `htdigest` 工具来创建和添加(纯文本)用户列表文件。

示例：

```
      <Location /private/>
  AuthType Digest
        AuthName "private area"
        AuthDigestDomain /private/ http://mirror.my.dom/private2/
    AuthDigestProvider file
        AuthUserFile /web/auth/.digest_pw
        Require valid-user
      </Location>
```

注意

摘要认证比基本认证更安全，但是直到2004年9月，只有下列最新版本的主流浏览器支持它：Amaya, Konqueror, MS Internet Explorer 6(使用查询字符串时会失败，参见"配合 MS Internet Explorer 6 工作"), Mozilla, Netscape 7, Opera, Safari 。而lynx不支持摘要认证。因

为摘要认证尚未得到绝大多数浏览器的支持，你应当只将它应用在你可以控制用户浏览器版本的场合。

配合 MS Internet Explorer 6 工作

Internet Explorer 6 的摘要认证实现有缺陷，也就是 `GET` 请求的查询字符串与RFC规范并不兼容。有几个途径来解决这个问题。

第一个途径就是使用 `POST` 代替 `GET` 来向服务器传送数据。如果你的程序不会受到这种变化的影响，这是最简单的方法。

从2.0.51版本开始，Apache还在环境变量 `AuthDigestEnableQueryStringHack` 中提供了一个工作区(workaround)。如果 `AuthDigestEnableQueryStringHack` 被打开，Apache将采取措施对付Internet Explorer 6 的bug，将请求URI从摘要比较中移除。使用这个方法将需要类似如下的配置：

在MSIE6中使用摘要认证

```
BrowserMatch "MSIE" AuthDigestEnableQueryStringHack=On
```

参见 `BrowserMatch` 指令以了解有条件的设置环境变量的更多细节。

AuthDigestAlgorithm 指令

说明	选择在摘要认证中用于计算请求和应答的散列值的算法
语法	`AuthDigestAlgorithm MD5|MD5-sess`
默认值	`AuthDigestAlgorithm MD5`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	实验(X)
模块	mod_auth_digest

`AuthDigestAlgorithm` 指令选择在摘要认证中用于计算请求和应答的散列值的算法。

`MD5-sess` 算法当前尚未实现。

AuthDigestDomain 指令

说明	在同一保护区域中需要进行摘要认证的URI
语法	`AuthDigestDomain URI [URI] ...`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	实验(X)
模块	mod_auth_digest

`AuthDigestDomain` 指令用于指定一个或者多个在同一保护区域中需要进行摘要认证的URI(也就是使用相同的区域和用户名/密码信息)。这些被指定的URI只是前缀，也就是说客户端将假定所有位于该URI"之下"的URI亦受到相同用户名/密码的保护。这些被指定的URI可以是绝对URI(也就是包含完整的协议、主机、端口等)或者相对URI。

这个指令必须总是被指定为至少包含被保护页面的根URI。省略这个会导致客户端为每个请求都发送授权头，除了增加请求的字节大小外，如果 `AuthDigestNcCheck` 被设为"On"，还会影响服务器的性能。

这里指定的URI可以分别指向不同的服务器，在这种情况下客户端将会在这些服务器间共享用户名和密码信息，并且不会提醒用户。

AuthDigestNcCheck 指令

说明	Enables or disables checking of the nonce-count sent by the server
语法	`AuthDigestNcCheck On|Off`
默认值	`AuthDigestNcCheck Off`
作用域	server config
状态	实验(X)
模块	mod_auth_digest

目前尚未实现。

AuthDigestNonceFormat 指令

说明	Determines how the nonce is generated
语法	`AuthDigestNonceFormat format`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	实验(X)
模块	mod_auth_digest

目前尚未实现。

AuthDigestNonceLifetime 指令

说明	服务器nonce(当前值)的有效秒数
语法	`AuthDigestNonceLifetime seconds`
默认值	`AuthDigestNonceLifetime 300`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	实验(X)
模块	mod_auth_digest

`AuthDigestNonceLifetime` 指令控制服务器nonce(当前值)的有效秒数。当客户端连接服务器时使用了一个过期的nonce(当前值)，服务器将返回一个带有" `stale=true` "的401错误(要求重新认证)。如果seconds小于等于"0"，那么nonce(当前值)将永远不会过期(强烈反对这么做)。一般这个值应当在30到120之间比较合理(最好不要小于10)。

AuthDigestProvider 指令

说明	设置该区域的(摘要)认证支持者(Provider)
语法	`AuthDigestProvider provider-name [provider-name] ...`
默认值	`AuthDigestProvider file`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	实验(X)
模块	mod_auth_digest

`AuthDigestProvider` 指令设置了该区域的(摘要)认证支持者(Provider)。默认的 `file` 支持者由 `mod_authn_file` 模块实现。必须确保所需的认证支持模块存在于服务器中(静态连接或DSO)。

能够提供认证支持者(Provider)的模块如下： `mod_authn_dbm` 和 `mod_authn_file` 。

AuthDigestQop 指令

说明	指定摘要认证的保护质量
语法	`AuthDigestQop none|auth|auth-int [auth|auth-int]`
默认值	`AuthDigestQop auth`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	实验(X)
模块	mod_auth_digest

`AuthDigestQop` 指令用于指定使用那个级别的<dfn class="calibre27">保护质量(quality-of-protection)</dfn>。 `auth` 将只进行认证(用户名/密码)； `auth-int` 除了认证以外还进行完整性校验(实体的MD5值将被计算和检查)； `none` 将使用旧的RFC-2069摘要算法(不包含完整性检查)； `auth` 和 `auth-int` 可以同时指定，在这种情况下，浏览器将会自己选择使用哪种一种。 `none` 不推荐使用。

`auth-int` 目前尚未支持。

AuthDigestShmemSize 指令

说明	为了跟踪客户端而分配的共享内存字节数
语法	`AuthDigestShmemSize size`
默认值	`AuthDigestShmemSize 1000`
作用域	server config
状态	实验(X)
模块	mod_auth_digest

`AuthDigestShmemSize` 指令指定了服务器启动时为了跟踪客户端而分配的共享内存字节数。注意，这个共享内存段不能设置为小于只跟踪一个客户端所需要的最小内存数量，这个最小数量取决于你的系统。如果你想知道这个最小值，你只要将 `AuthDigestShmemSize` 设为" `0` "，然后读取重启Apache时返回的错误信息即可。

size通常按照字节计算，但是可以通过加上后缀"`K`"或"`M`"来按照KB或MB计算。比如，以下写法都是一样的：

```
AuthDigestShmemSize 1048576

AuthDigestShmemSize 1024K

AuthDigestShmemSize 1M
```

Apache模块 mod_authn_alias

说明	基于实际认证支持者创建扩展的认证支持者，并为它起一个别名以便于引用
状态	扩展(E)
模块名	authn_alias_module
源文件	mod_authn_alias.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

这个模块可以在配置文件中基于实际认证支持者创建扩展的认证支持者，并为它起一个别名以便于在 `AuthBasicProvider` 或 `AuthDigestProvider` 指令中像实际认证支持者一样被引用。此外，它还允许同一个认证支持者在不同的区域被多次引用。

示例

下面的例子创建了基于ldap(实际)认证支持者的两个不同的ldap(扩展)认证支持者别名。这样，同一个认证区域就可以被多个ldap主机伺服。

示例

```
LoadModule authn_alias_module modules/mod_authn_alias.so

<AuthnProviderAlias ldap ldap-alias1>

AuthLDAPBindDN cn=youruser,o=ctx

AuthLDAPBindPassword yourpassword

AuthLDAPURL ldap://ldap.host/o=ctx
</AuthnProviderAlias>

<AuthnProviderAlias ldap ldap-other-alias>

AuthLDAPBindDN cn=yourotheruser,o=dev

AuthLDAPBindPassword yourotherpassword

AuthLDAPURL ldap://other.ldap.host/o=dev?cn
</AuthnProviderAlias>

Alias /secure /webpages/secure

<Directory /webpages/secure>

Order deny,allow

Allow from all

AuthBasicProvider ldap-other-alias  ldap-alias1

AuthType Basic

AuthName LDAP_Protected_Place

AuthzLDAPAuthoritative off

require valid-user
</Directory>
```

<AuthnProviderAlias> 指令

说明	封装一组定义扩展认证支持者的指令，并为其指定一个别名
语法	`<AuthnProviderAlias baseProvider Alias> ... </AuthnProviderAlias>`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_authn_alias

`<AuthnProviderAlias>` 和 `</AuthnProviderAlias>` 用来封装一组定义扩展认证支持者的指令，并为其指定一个别名，这个别名可以被 `AuthBasicProvider` 或 `AuthDigestProvider` 引用。

Apache模块 mod_authn_anon

说明	提供匿名用户认证支持
状态	扩展(E)
模块名	authn_anon_module
源文件	mod_authn_anon.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

This module provides authentication front-ends such as `mod_auth_basic` to authenticate users similar to anonymous-ftp sites, *i.e.* have a 'magic' user id 'anonymous' and the email address as a password. These email addresses can be logged.

Combined with other (database) access control methods, this allows for effective user tracking and customization according to a user profile while still keeping the site open for 'unregistered' users. One advantage of using Auth-based user tracking is that, unlike magic-cookies and funny URL pre/postfixes, it is completely browser independent and it allows users to share URLs.

When using `mod_auth_basic` , this module is invoked via the `AuthBasicProvider` directive with the `anon` value.

Example

The example below is combined with "normal" htpasswd-file based authentication and allows users in additionally as 'guests' with the following properties:

- It insists that the user enters a userID. (`Anonymous_NoUserID`)
- It insists that the user enters a password. (`Anonymous_MustGiveEmail`)
- The password entered must be a valid email address, *i.e.* contain at least one '@' and a '.'. (`Anonymous_VerifyEmail`)
- The userID must be one of `anonymous guest www test welcome` and comparison is **not** case sensitive. (`Anonymous`)
- And the Email addresses entered in the passwd field are logged to the error log file. (`Anonymous_LogEmail`)

示例

```
<Directory /foo>
AuthName "Use 'anonymous' & Email address for guest entry"

  AuthType Basic

  AuthBasicProvider file anon

  AuthUserFile /path/to/your/.htpasswd

  Anonymous_NoUserID off

  Anonymous_MustGiveEmail on

  Anonymous_VerifyEmail on

  Anonymous_LogEmail on

  Anonymous anonymous guest www test welcome

  Order Deny,Allow

  Allow from all

  Require valid-user
</Directory>
```

Anonymous 指令

说明	**Specifies userIDs that are allowed access without password verification**
语法	`Anonymous user [user] ...`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authn_anon

A list of one or more 'magic' userIDs which are allowed access without password verification. The userIDs are space separated. It is possible to use the ' and " quotes to allow a space in a userID as well as the \ escape character.

Please note that the comparison is **case-IN-sensitive**. It's strongly recommended that the magic username '`anonymous`' is always one of the allowed userIDs.

示例：

```
Anonymous anonymous "Not Registered" "I don't know"
```

This would allow the user to enter without password verification by using the userIDs "anonymous", "AnonyMous", "Not Registered" and "I Don't Know".

As of Apache 2.1 it is possible to specify the userID as " `*` ". That allows *any* supplied userID to be accepted.

Anonymous_LogEmail 指令

说明	Sets whether the password entered will be logged in the error log
语法	`Anonymous_LogEmail On|Off`
默认值	`Anonymous_LogEmail On`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authn_anon

When set `On` , the default, the 'password' entered (which hopefully contains a sensible email address) is logged in the error log.

Anonymous_MustGiveEmail 指令

说明	Specifies whether blank passwords are allowed
语法	`Anonymous_MustGiveEmail On|Off`
默认值	`Anonymous_MustGiveEmail On`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authn_anon

Specifies whether the user must specify an email address as the password. This prohibits blank passwords.

Anonymous_NoUserID 指令

说明	Sets whether the userID field may be empty
语法	`Anonymous_NoUserID On|Off`
默认值	`Anonymous_NoUserID Off`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authn_anon

When set `On` , users can leave the userID (and perhaps the password field) empty. This can be very convenient for MS-Explorer users who can just hit return or click directly on the OK button; which seems a natural reaction.

Anonymous_VerifyEmail 指令

说明	Sets whether to check the password field for a correctly formatted email address
语法	`Anonymous_VerifyEmail On|Off`
默认值	`Anonymous_VerifyEmail Off`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authn_anon

When set `On` the 'password' entered is checked for at least one '@' and a '.' to encourage users to enter valid email addresses (see the above `Anonymous_LogEmail`).

Apache模块 mod_authn_dbd

说明	使用SQL数据库为认证提供支持
状态	扩展(E)
模块名	authn_dbd_module
源文件	mod_authn_dbd.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

This module provides authentication front-ends such as `mod_auth_digest` 和 `mod_auth_basic` to authenticate users by looking up users in SQL tables. Similar functionality is provided by, for example, `mod_authn_file` .

This module relies on `mod_dbd` to specify the backend database driver and connection parameters, and manage the database connections.

When using `mod_auth_basic` 或 `mod_auth_digest` , this module is invoked via the `AuthBasicProvider` 或 `AuthDigestProvider` with the `dbd` value.

Configuration Example

This simple example shows use of this module in the context of the Authentication and DBD frameworks.

```
#Database Management

#Use the PostgreSQL driver
DBDriver pgsql

#Connection string: database name and login credentials
DBDParams "dbname=htpasswd user=apache pass=xxxxxx"

#Parameters for Connection Pool Management
DBDMin  1
DBDKeep 2
DBDMax  10
DBDExptime 60

#Authentication Section
<Directory /usr/www/myhost/private>

    #mod_auth configuration for authn_dbd
    AuthType Basic
    AuthName "My Server"
    AuthBasicProvider dbd

    #authz configuration
    Require valid-user

    #SQL query to verify a user
    #(note: DBD drivers recognise both stdio-like %s and native syntax)
    AuthDBDUserPWQuery "select password from authn where username = %s"
</Directory>
```

AuthDBDUserPWQuery 指令

说明	SQL query to look up a password for a user
语法	`AuthDBDUserPWQuery query`
作用域	directory
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authn_dbd

`AuthDBDUserPWQuery` specifies an SQL query to look up a password for a specified user. The query must take a single string (typically SQL varchar) argument (username), and return a single value (encrypted password).

```
    AuthDBDUserPWQuery "SELECT password FROM authn WHERE username = %s"
```

AuthDBDUserRealmQuery 指令

说明	SQL query to look up a password hash for a user and realm.
语法	`AuthDBDUserRealmQuery query`
作用域	directory
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authn_dbd

`AuthDBDUserRealmPWQuery` specifies an SQL query to look up a password for a specified user and realm. The query must take two string (typically SQL varchar) arguments (username and realm), and return a single value (encrypted password).

```
AuthDBDUserRealmPWQuery "SELECT password FROM authn
                                WHERE username = %s AND realm = %s"
```

Apache模块 mod_authn_dbm

说明	使用DBM数据库为认证提供支持
状态	扩展(E)
模块名	authn_dbm_module
源文件	mod_authn_dbm.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

This module provides authentication front-ends such as `mod_auth_digest` 和 `mod_auth_basic` to authenticate users by looking up users in <dfn class="calibre27">dbm</dfn> password files. Similar functionality is provided by `mod_authn_file` .

When using `mod_auth_basic` 或 `mod_auth_digest` , this module is invoked via the `AuthBasicProvider` 或 `AuthDigestProvider` with the `dbm` value.

AuthDBMType 指令

说明	Sets the type of database file that is used to store passwords
语法	`AuthDBMType default|SDBM|GDBM|NDBM|DB`
默认值	`AuthDBMType default`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authn_dbm

Sets the type of database file that is used to store the passwords. The default database type is determined at compile time. The availability of other types of database files also depends on compile-time settings.

It is crucial that whatever program you use to create your password files is configured to use the same type of database.

AuthDBMUserFile 指令

说明	**Sets the name of a database file containing the list of users and passwords for authentication**
语法	`AuthDBMUserFile file-path`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authn_dbm

`AuthDBMUserFile` directive sets the name of a DBM file containing the list of users and passwords for user authentication. File-path is the absolute path to the user file.

The user file is keyed on the username. The value for a user is the encrypted password, optionally followed by a colon and arbitrary data. The colon and the data following it will be ignored by the server.

安全

Make sure that the `AuthDBMUserFile` is stored outside the document tree of the web-server; do *not* put it in the directory that it protects. Otherwise, clients will be able to download the `AuthDBMUserFile` .

Important compatibility note: The implementation of `dbmopen` in the apache modules reads the string length of the hashed values from the DBM data structures, rather than relying upon the string being NULL-appended. Some applications, such as the Netscape web server, rely upon the string being NULL-appended, so if you are having trouble using DBM files interchangeably between applications this may be a part of the problem.

A perl script called `dbmmanage` is included with Apache. This program can be used to create and update DBM format password files for use with this module.

Apache模块 mod_authn_default

说明	在未正确配置认证模块的情况下简单拒绝一切认证信息
状态	基本(B)
模块名	authn_default_module
源文件	mod_authn_default.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

该模块是一个失败补救(fallback)模块，它在未正确配置认证模块(比如 `mod_auth_basic`)的情况下简单拒绝一切认证信息。

AuthDefaultAuthoritative 指令

说明	指定是否将认证操作交由更底层的模块来处理
语法	`AuthDefaultAuthoritative On|Off`
默认值	`AuthDefaultAuthoritative On`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	基本(B)
模块	mod_authn_default

将 `AuthDefaultAuthoritative` 明确设置为 `Off` 将允许将认证操作交由更底层的(在 `modules.c` 文件中定义的)模块来处理。

注意

通常并不存在更底层的模块，因为 `mod_authn_default` 已经被定义为非常底层的模块了。因此最好将 `AuthDefaultAuthoritative` 保持其默认值(`On`)。

Apache模块 mod_authn_file

说明	使用纯文本文件为认证提供支持
状态	基本(B)
模块名	authn_file_module
源文件	mod_authn_file.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

该模块为认证前端(`mod_auth_digest` 和 `mod_auth_basic`)使用纯文本文件进行用户认证提供支持。 `mod_authn_dbm` 模块也提供类似的功能。

使用 `mod_auth_basic` 或 `mod_auth_digest` 的时候，可以通过在 `AuthBasicProvider` 或 `AuthDigestProvider` 指令中使用 `file` 值调用该模块。

AuthUserFile 指令

说明	设定一个含有认证使用的用户名/密码列表的纯文本文件
语法	`AuthUserFile file-path`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	基本(B)
模块	mod_authn_file

`AuthUserFile` 指令设定一个纯文本文件的名称，其中包含用于认证的用户名/密码的列表，File-path是该文件的路径。如果不是绝对路径(也就是说，如果不是以斜杠开始的)，则是相对于 `ServerRoot` 的相对路径。

用户文件的每一行包含一个用户名，后跟一个冒号，再跟一个加密过的密码。如果文件中含有重复的用户名， `mod_authn_file` 模块会用排在最前面的那行定义来验证该用户的密码。

在二进制文件安装包中附带的(或者在" `src/support` "中)命令行工具 `htpasswd` 可以用来维护仅用于*HTTP*基本认证的密码文件。参阅手册页面以获得更详细的说明。

以一个初始帐户 `username` 创建一个密码文件 `Filename` 。它会提示输入密码：

```
htpasswd -c Filename username
```

增加或修改密码文件 `Filename` 中的帐号 `username2` ：

```
htpasswd Filename username2
```

注意：搜索很大的文本文件是非常慢的；应该使用 `AuthDBMUserFile` 来替代它。

如果使用*HTTP*摘要认证，就不能使用 `htpasswd` 工具，而要使用 `htdigest` 工具。注意：不能在同一个文件中同时包含用于基本认证和摘要认证的用户数据。

安全

必须确保 `AuthUserFile` 文件存放在WEB服务器目录之外，千万不要放在它所保护的目录中，否则可能会被客户端下载。

Apache模块 mod_authnz_ldap

说明	允许使用一个LDAP目录存储用户名和密码数据库来执行基本认证和授权
状态	扩展(E)
模块名	authnz_ldap_module
源文件	mod_authnz_ldap.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

This module provides authentication front-ends such as `mod_auth_basic` to authenticate users through an ldap directory.

`mod_authnz_ldap` supports the following features:

- Known to support the OpenLDAP SDK (both 1.x and 2.x), Novell LDAP SDK and the iPlanet (Netscape) SDK.
- Complex authorization policies can be implemented by representing the policy with LDAP filters.
- Uses extensive caching of LDAP operations via mod_ldap.
- Support for LDAP over SSL (requires the Netscape SDK) or TLS (requires the OpenLDAP 2.x SDK or Novell LDAP SDK).

When using `mod_auth_basic` , this module is invoked via the `AuthBasicProvider` directive with the `ldap` value.

Contents

Operation

There are two phases in granting access to a user. The first phase is authentication, in which the `mod_authnz_ldap` authentication provider verifies that the user's credentials are valid. This is also called the *search/bind* phase. The second phase is authorization, in which `mod_authnz_ldap` determines if the authenticated user is allowed access to the resource in question. This is also known as the *compare* phase.

`mod_authnz_ldap` registers both an authn_ldap authentication provider and an authz_ldap authorization handler. The authn_ldap authentication provider can be enabled through the `AuthBasicProvider` directive using the `ldap` value. The authz_ldap handler extends the `Require` directive's authorization types by adding `ldap-user` , `ldap-dn` 和 `ldap-group` values.

The Authentication Phase

During the authentication phase, `mod_authnz_ldap` searches for an entry in the directory that matches the username that the HTTP client passes. If a single unique match is found, then `mod_authnz_ldap` attempts to bind to the directory server using the DN of the entry plus the password provided by the HTTP client. Because it does a search, then a bind, it is often referred to as the search/bind phase. Here are the steps taken during the search/bind phase.

1. Generate a search filter by combining the attribute and filter provided in the `AuthLDAPURL` directive with the username passed by the HTTP client.
2. Search the directory using the generated filter. If the search does not return exactly one entry, deny or decline access.
3. Fetch the distinguished name of the entry retrieved from the search and attempt to bind to the LDAP server using the DN and the password passed by the HTTP client. If the bind is unsuccessful, deny or decline access.

The following directives are used during the search/bind phase

`AuthLDAPURL`	**Specifies the LDAP server, the base DN, the attribute to use in the search, as well as the extra search filter to use.**
`AuthLDAPBindDN`	An optional DN to bind with during the search phase.
`AuthLDAPBindPassword`	An optional password to bind with during the search phase.

The Authorization Phase

During the authorization phase, `mod_authnz_ldap` attempts to determine if the user is authorized to access the resource. Many of these checks require `mod_authnz_ldap` to do a compare operation on the LDAP server. This is why this phase is often referred to as the compare phase. `mod_authnz_ldap` accepts the following `Require` directives to determine if the credentials are acceptable:

- Grant access if there is a `require ldap-user` directive, and the username in the directive matches the username passed by the client.
- Grant access if there is a `require ldap-dn` directive, and the DN in the directive matches the DN fetched from the LDAP directory.
- Grant access if there is a `require ldap-group` directive, and the DN fetched from the LDAP directory (or the username passed by the client) occurs in the LDAP group.
- Grant access if there is a `require ldap-attribute` directive, and the attribute fetched from the LDAP directory matches the given value.
- Grant access if there is a `require ldap-filter` directive, and the search filter successfully finds a single user object that matches the dn of the authenticated user.
- otherwise, deny or decline access

Other `Require` values may also be used which may require loading additional authorization modules.

- Grant access if there is a `require valid-user` directive. (requires `mod_authz_user`)
- Grant access if there is a `require group` directive, and `mod_authz_groupfile` has been loaded with the `AuthGroupFile` directive set.
- others...

`mod_authnz_ldap` uses the following directives during the compare phase:

`AuthLDAPURL`	**The attribute specified in the URL is used in compare operations for the `require ldap-user` operation.**
`AuthLDAPCompareDNOnServer`	Determines the behavior of the `require ldap-dn` directive.
`AuthLDAPGroupAttribute`	Determines the attribute to use for comparisons in the `require ldap-group` directive.
`AuthLDAPGroupAttributeIsDN`	Specifies whether to use the user DN or the username when doing comparisons for the `require ldap-group` directive.

The require Directives

Apache's `Require` directives are used during the authorization phase to ensure that a user is allowed to access a resource. mod_authnz_ldap extends the authorization types with `ldap-user` , `ldap-dn` , `ldap-group` , `ldap-attribute` 和 `ldap-filter` . Other authorization types may also be used but may require that additional authorization modules be loaded.

require valid-user

If this directive exists, `mod_authnz_ldap` grants access to any user that has successfully authenticated during the search/bind phase. Requires that `mod_authz_user` be loaded and that the `AuthzLDAPAuthoritative` directive be set to off.

require ldap-user

`require ldap-user` directive specifies what usernames can access the resource. Once `mod_authnz_ldap` has retrieved a unique DN from the directory, it does an LDAP compare operation using the username specified in the `require ldap-user` to see if that username is part of the just-fetched LDAP entry. Multiple users can be granted access by putting multiple usernames on the line, separated with spaces. If a username has a space in it, then it must be surrounded with double quotes. Multiple users can also be granted access by using multiple `require ldap-user` directives, with one user per line. For example, with a `AuthLDAPURL` of `ldap://ldap/o=Airius?cn` (i.e., `cn` is used for searches), the following require directives could be used to restrict access:

```
require ldap-user "Barbara Jenson"

require ldap-user "Fred User"

require ldap-user "Joe Manager"
```

Because of the way that `mod_authnz_ldap` handles this directive, Barbara Jenson could sign on as *Barbara Jenson*, *Babs Jenson* or any other `cn` that she has in her LDAP entry. Only the single `require ldap-user` line is needed to support all values of the attribute in the user's entry.

If the `uid` attribute was used instead of the `cn` attribute in the URL above, the above three lines could be condensed to

```
require ldap-user bjenson fuser jmanager
```

require ldap-group

This directive specifies an LDAP group whose members are allowed access. It takes the distinguished name of the LDAP group. Note: Do not surround the group name with quotes. For example, assume that the following entry existed in the LDAP directory:

```
dn: cn=Administrators, o=Airius

objectClass: groupOfUniqueNames

uniqueMember: cn=Barbara Jenson, o=Airius

uniqueMember: cn=Fred User, o=Airius
```

The following directive would grant access to both Fred and Barbara:

```
require ldap-group cn=Administrators, o=Airius
```

Behavior of this directive is modified by the `AuthLDAPGroupAttribute` 和 `AuthLDAPGroupAttributeIsDN` directives.

require ldap-dn

`require ldap-dn` directive allows the administrator to grant access based on distinguished names. It specifies a DN that must match for access to be granted. If the distinguished name that was retrieved from the directory server matches the distinguished name in the `require ldap-dn` , then authorization is granted. Note: do not surround the distinguished name with quotes.

The following directive would grant access to a specific DN:

```
require ldap-dn cn=Barbara Jenson, o=Airius
```

Behavior of this directive is modified by the `AuthLDAPCompareDNOnServer` directive.

require ldap-attribute

`require ldap-attribute` directive allows the administrator to grant access based on attributes of the authenticated user in the LDAP directory. If the attribute in the directory matches the value given in the configuration, access is granted.

The following directive would grant access to anyone with the attribute employeeType = active

```
require ldap-attribute employeeType=active
```

Multiple attribute/value pairs can be specified on the same line separated by spaces or they can be specified in multiple `require ldap-attribute` directives. The effect of listing multiple attribute/values pairs is an OR operation. Access will be granted if any of the listed attribute values match the value of the corresponding attribute in the user object. If the value of the attribute contains a space, only the value must be within double quotes.

The following directive would grant access to anyone with the city attribute equal to "San Jose" or status equal to "Active"

```
require ldap-attribute city="San Jose" status=active
```

require ldap-filter

`require ldap-filter` directive allows the administrator to grant access based on a complex LDAP search filter. If the dn returned by the filter search matches the authenticated user dn, access is granted.

The following directive would grant access to anyone having a cell phone and is in the marketing department

```
require ldap-filter &(cell=*)(department=marketing)
```

The difference between the `require ldap-filter` directive and the `require ldap-attribute` directive is that `ldap-filter` performs a search operation on the LDAP directory using the specified search filter rather than a simple attribute comparison. If a simple attribute comparison is all that is required, the comparison operation performed by `ldap-attribute` will be faster than the search operation used by `ldap-filter` especially within a large directory.

Examples

- Grant access to anyone who exists in the LDAP directory, using their UID for searches.

```
AuthLDAPURL ldap://ldap1.airius.com:389/ou=People, o=Airius?uid?sub?(objectClass=*)
require valid-user
```

- The next example is the same as above; but with the fields that have useful defaults omitted. Also, note the use of a redundant LDAP server.

```
AuthLDAPURL ldap://ldap1.airius.com ldap2.airius.com/ou=People, o=Airius
require valid-user
```

- The next example is similar to the previous one, but it uses the common name instead of the UID. Note that this could be problematical if multiple people in the directory share the same `cn`, because a search on `cn` **must** return exactly one entry. That's why this approach is not recommended: it's a better idea to choose an attribute that is guaranteed unique in your directory, such as `uid`.

```
AuthLDAPURL ldap://ldap.airius.com/ou=People, o=Airius?cn
require valid-user
```

- Grant access to anybody in the Administrators group. The users must authenticate using their UID.

```
AuthLDAPURL ldap://ldap.airius.com/o=Airius?uid
require ldap-group cn=Administrators, o=Airius
```

- The next example assumes that everyone at Airius who carries an alphanumeric pager will have an LDAP attribute of `qpagePagerID`. The example will grant access only to people (authenticated via their UID) who have alphanumeric pagers:

```
AuthLDAPURL ldap://ldap.airius.com/o=Airius?uid??(qpagePagerID=*)
require valid-user
```

- The next example demonstrates the power of using filters to accomplish complicated administrative requirements. Without filters, it would have been necessary to create a new LDAP group and ensure that the group's members remain synchronized with the pager users. This becomes trivial with filters. The goal is to grant access to anyone who has a pager, plus grant access to Joe Manager, who doesn't have a pager, but does need to access the same resource:

```
AuthLDAPURL ldap://ldap.airius.com/o=Airius?uid??(|(qpagePagerID=*)(uid=jmanager))
require valid-user
```

This last may look confusing at first, so it helps to evaluate what the search filter will look like based on who connects, as shown below. If Fred User connects as `fuser`, the filter would look like

```
(&(|(qpagePagerID=*)(uid=jmanager))(uid=fuser))
```

The above search will only succeed if *fuser* has a pager. When Joe Manager connects as *jmanager*, the filter looks like

```
(&(|(qpagePagerID=*)(uid=jmanager))(uid=jmanager))
```

The above search will succeed whether *jmanager* has a pager or not.

Using TLS

To use TLS, see the `mod_ldap` directives `LDAPTrustedClientCert`, `LDAPTrustedGlobalCert` 和 `LDAPTrustedMode`.

An optional second parameter can be added to the `AuthLDAPURL` to override the default connection type set by `LDAPTrustedMode`. This will allow the connection established by an *ldap://* Url to be upgraded to a secure connection on the same port.

Using SSL

To use SSL, see the `mod_ldap` directives `LDAPTrustedClientCert`, `LDAPTrustedGlobalCert` 和 `LDAPTrustedMode`.

To specify a secure LDAP server, use *ldaps://* in the `AuthLDAPURL` directive, instead of *ldap://*.

Using Microsoft FrontPage with mod_authnz_ldap

Normally, FrontPage uses FrontPage-web-specific user/group files (i.e., the `mod_authn_file` 和 `mod_authz_groupfile` modules) to handle all authentication. Unfortunately, it is not possible to just change to LDAP authentication by adding the proper directives,

because it will break the *Permissions* forms in the FrontPage client, which attempt to modify the standard text-based authorization files.

Once a FrontPage web has been created, adding LDAP authentication to it is a matter of adding the following directives to *every* `.htaccess` file that gets created in the web

```
AuthLDAPURL            "the url"
AuthzLDAPAuthoritative off
AuthGroupFile _mygroupfile_
require group _mygroupfile_
```

`AuthzLDAPAuthoritative` must be off to allow `mod_authnz_ldap` to decline group authentication so that Apache will fall back to file authentication for checking group membership. This allows the FrontPage-managed group file to be used.

How It Works

FrontPage restricts access to a web by adding the `require valid-user` directive to the `.htaccess` files. The `require valid-user` directive will succeed for any user who is valid *as far as LDAP is concerned*. This means that anybody who has an entry in the LDAP directory is considered a valid user, whereas FrontPage considers only those people in the local user file to be valid. By substituting the ldap-group with group file authorization, Apache is allowed to consult the local user file (which is managed by FrontPage) - instead of LDAP - when handling authorizing the user.

Once directives have been added as specified above, FrontPage users will be able to perform all management operations from the FrontPage client.

Caveats

- When choosing the LDAP URL, the attribute to use for authentication should be something that will also be valid for putting into a `mod_authn_file` user file. The user ID is ideal for this.
- When adding users via FrontPage, FrontPage administrators should choose usernames that already exist in the LDAP directory (for obvious reasons). Also, the password that the administrator enters into the form is ignored, since Apache will actually be authenticating against the password in the LDAP database, and not against the password in the local user file. This could cause confusion for web administrators.
- Apache must be compiled with `mod_auth_basic` , `mod_authn_file` 和 `mod_authz_groupfile` in order to use FrontPage support. This is because Apache will still use the `mod_authz_groupfile` group file for determine the extent of a user's access to the FrontPage web.
- The directives must be put in the `.htaccess` files. Attempting to put them inside

`<Location>` 或 `<Directory>` directives won't work. This is because `mod_authnz_ldap` has to be able to grab the `AuthGroupFile` directive that is found in FrontPage `.htaccess` files so that it knows where to look for the valid user list. If the `mod_authnz_ldap` directives aren't in the same `.htaccess` file as the FrontPage directives, then the hack won't work, because `mod_authnz_ldap` will never get a chance to process the `.htaccess` file, and won't be able to find the FrontPage-managed user file.

AuthLDAPBindDN 指令

说明	Optional DN to use in binding to the LDAP server
语法	`AuthLDAPBindDN _distinguished-name_`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authnz_ldap

An optional DN used to bind to the server when searching for entries. If not provided, `mod_authnz_ldap` will use an anonymous bind.

AuthLDAPBindPassword 指令

说明	Password used in conjuction with the bind DN
语法	`AuthLDAPBindPassword _password_`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authnz_ldap

A bind password to use in conjunction with the bind DN. Note that the bind password is probably sensitive data, and should be properly protected. You should only use the `AuthLDAPBindDN` 和 `AuthLDAPBindPassword` if you absolutely need them to search the directory.

AuthLDAPCharsetConfig 指令

说明	Language to charset conversion configuration file
语法	`AuthLDAPCharsetConfig _file-path_`
作用域	server config
状态	扩展(E)
模块	mod_authnz_ldap

`AuthLDAPCharsetConfig` directive sets the location of the language to charset conversion configuration file. File-path is relative to the `ServerRoot`. This file specifies the list of language extensions to character sets. Most administrators use the provided `charset.conv` file, which associates common language extensions to character sets.

The file contains lines in the following format:

```
<var class="calibre40">Language-Extension</var> <var class="calibre40">charset</var> [<va
```

The case of the extension does not matter. Blank lines, and lines beginning with a hash character (`#`) are ignored.

AuthLDAPCompareDNOnServer 指令

说明	Use the LDAP server to compare the DNs
语法	`AuthLDAPCompareDNOnServer on|off`
默认值	`AuthLDAPCompareDNOnServer on`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authnz_ldap

When set, `mod_authnz_ldap` will use the LDAP server to compare the DNs. This is the only foolproof way to compare DNs. `mod_authnz_ldap` will search the directory for the DN specified with the `require dn` directive, then, retrieve the DN and compare it with the DN retrieved from the user entry. If this directive is not set, `mod_authnz_ldap` simply does a string comparison. It is possible to get false negatives with this approach, but it is much faster. Note the `mod_ldap` cache can speed up DN comparison in most situations.

AuthLDAPDereferenceAliases 指令

说明	When will the module de-reference aliases
语法	`AuthLDAPDereferenceAliases never|searching|finding|always`
默认值	`AuthLDAPDereferenceAliases Always`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authnz_ldap

This directive specifies when `mod_authnz_ldap` will de-reference aliases during LDAP operations. The default is `always` .

AuthLDAPGroupAttribute 指令

说明	LDAP attributes used to check for group membership
语法	`AuthLDAPGroupAttribute _attribute_`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authnz_ldap

This directive specifies which LDAP attributes are used to check for group membership. Multiple attributes can be used by specifying this directive multiple times. If not specified, then `mod_authnz_ldap` uses the `member` 和 `uniquemember` attributes.

AuthLDAPGroupAttributeIsDN 指令

说明	Use the DN of the client username when checking for group membership
语法	`AuthLDAPGroupAttributeIsDN on|off`
默认值	`AuthLDAPGroupAttributeIsDN on`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authnz_ldap

When set `on`, this directive says to use the distinguished name of the client username when checking for group membership. Otherwise, the username will be used. For example, assume that the client sent the username `bjenson`, which corresponds to the LDAP DN `cn=Babs Jenson, o=Airius`. If this directive is set, `mod_authnz_ldap` will check if the group has `cn=Babs Jenson, o=Airius` as a member. If this directive is not set, then `mod_authnz_ldap` will check if the group has `bjenson` as a member.

AuthLDAPRemoteUserIsDN 指令

说明	Use the DN of the client username to set the REMOTE_USER environment variable
语法	`AuthLDAPRemoteUserIsDN on|off`
默认值	`AuthLDAPRemoteUserIsDN off`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authnz_ldap

If this directive is set to on, the value of the `REMOTE_USER` environment variable will be set to the full distinguished name of the authenticated user, rather than just the username that was passed by the client. It is turned off by default.

AuthLDAPUrl 指令

说明	URL specifying the LDAP search parameters
语法	`AuthLDAPUrl _url [NONE|SSL|TLS|STARTTLS]_`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authnz_ldap

An RFC 2255 URL which specifies the LDAP search parameters to use. The syntax of the URL is

```
ldap://host:port/basedn?attribute?scope?filter
```

ldap

For regular ldap, use the string `ldap`. For secure LDAP, use `ldaps` instead. Secure LDAP is only available if Apache was linked to an LDAP library with SSL support.

host:port

The name/port of the ldap server (defaults to `localhost:389` for `ldap`, and `localhost:636` for `ldaps`). To specify multiple, redundant LDAP servers, just list all servers, separated by spaces. `mod_authnz_ldap` will try connecting to each server in turn, until it makes a successful connection.

Once a connection has been made to a server, that connection remains active for the life of the `httpd` process, or until the LDAP server goes down.

If the LDAP server goes down and breaks an existing connection, `mod_authnz_ldap` will attempt to re-connect, starting with the primary server, and trying each redundant server in turn. Note that this is different than a true round-robin search.

basedn

The DN of the branch of the directory where all searches should start from. At the very least, this must be the top of your directory tree, but could also specify a subtree in the directory.

attribute

The attribute to search for. Although RFC 2255 allows a comma-separated list of attributes, only the first attribute will be used, no matter how many are provided. If no attributes are provided, the default is to use `uid`. It's a good idea to choose an attribute that will be unique across all entries in the subtree you will be using.

scope

The scope of the search. Can be either `one` 或 `sub` . Note that a scope of `base` is also supported by RFC 2255, but is not supported by this module. If the scope is not provided, or if `base` scope is specified, the default is to use a scope of `sub` .

filter

A valid LDAP search filter. If not provided, defaults to `(objectClass=*)` , which will search for all objects in the tree. Filters are limited to approximately 8000 characters (the definition of `MAX_STRING_LEN` in the Apache source code). This should be than sufficient for any application.

When doing searches, the attribute, filter and username passed by the HTTP client are combined to create a search filter that looks like `(&(_filter_)(_attribute_=_username_))` .

For example, consider an URL of `ldap://ldap.airius.com/o=Airius?cn?sub?(posixid=*)` . When a client attempts to connect using a username of `Babs Jenson` , the resulting search filter will be `(&(posixid=*)(cn=Babs Jenson))` .

An optional parameter can be added to allow the LDAP Url to override the connection type. This parameter can be one of the following:

NONE

Establish an unsecure connection on the default LDAP port. This is the same as `ldap://` on port 389.

SSL

Establish a secure connection on the default secure LDAP port. This is the same as `ldaps://`

TLS | STARTTLS

Establish an upgraded secure connection on the default LDAP port. This connection will be initiated on port 389 by default and then upgraded to a secure connection on the same port.

See above for examples of `AuthLDAPURL` URLs.

AuthzLDAPAuthoritative 指令

说明	**Prevent other authentication modules from authenticating the user if this one fails**
语法	`AuthzLDAPAuthoritative on|off`
默认值	`AuthzLDAPAuthoritative on`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authnz_ldap

Set to `off` if this module should let other authentication modules attempt to authenticate the user, should authentication with this module fail. Control is only passed on to lower modules if there is no DN or rule that matches the supplied user name (as passed by the client).

Apache模块 mod_authz_dbm

说明	使用DBM数据库文件为组提供授权支持
状态	扩展(E)
模块名	authz_dbm_module
源文件	mod_authz_dbm.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

This module provides authorization capabilities so that authenticated users can be allowed or denied access to portions of the web site by group membership. Similar functionality is provided by `mod_authz_groupfile` .

AuthDBMGroupFile 指令

说明	Sets the name of the database file containing the list of user groups for authorization
语法	`AuthDBMGroupFile file-path`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authz_dbm

`AuthDBMGroupFile` directive sets the name of a DBM file containing the list of user groups for user authorization. File-path is the absolute path to the group file.

The group file is keyed on the username. The value for a user is a comma-separated list of the groups to which the users belongs. There must be no whitespace within the value, and it must never contain any colons.

安全

Make sure that the `AuthDBMGroupFile` is stored outside the document tree of the web-server. Do **not** put it in the directory that it protects. Otherwise, clients will be able to download the `AuthDBMGroupFile` unless otherwise protected.

Combining Group and Password DBM files: In some cases it is easier to manage a single database which contains both the password and group details for each user. This simplifies any support programs that need to be written: they now only have to deal with writing to and locking a single DBM file. This can be accomplished by first setting the group and password files to point to the same DBM:

```
AuthDBMGroupFile /www/userbase

AuthDBMUserFile /www/userbase
```

The key for the single DBM is the username. The value consists of

```
Encrypted Password : List of Groups [ : (ignored) ]
```

The password section contains the encrypted password as before. This is followed by a colon and the comma separated list of groups. Other data may optionally be left in the DBM file after another colon; it is ignored by the authorization module. This is what www.telescope.org uses for its combined password and group database.

AuthzDBMAuthoritative 指令

说明	**Sets whether authorization will be passed on to lower level modules**
语法	`AuthzDBMAuthoritative On|Off`
默认值	`AuthzDBMAuthoritative On`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authz_dbm

Setting the `AuthzDBMAuthoritative` directive explicitly to `Off` allows group authorization to be passed on to lower level modules (as defined in the `modules.c` file) if there is no group found for the the supplied userID. If there are any groups specified, the usual checks will be applied and a failure will give an Authentication Required reply.

So if a userID appears in the database of more than one module; or if a valid `Require` directive applies to more than one module; then the first module will verify the credentials; and no access is passed on; regardless of the `AuthBasicAuthoritative` setting.

A common use for this is in conjunction with one of the auth providers; such as `mod_authn_dbm` 或 `mod_authn_file` . Whereas this DBM module supplies the bulk of the user credential checking; a few (administrator) related accesses fall through to a lower level with a well protected `.htpasswd` file.

By default, control is not passed on and an unknown group will result in an Authentication Required reply. Not setting it thus keeps the system secure and forces an NCSA compliant behaviour.

安全

Do consider the implications of allowing a user to allow fall-through in his .htaccess file; and verify that this is really what you want; Generally it is easier to just secure a single `.htpasswd` file, than it is to secure a database which might have more access interfaces.

AuthzDBMType 指令

说明	Sets the type of database file that is used to store list of user groups
语法	`AuthzDBMType default|SDBM|GDBM|NDBM|DB`
默认值	`AuthzDBMType default`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authz_dbm

Sets the type of database file that is used to store the list of user groups. The default database type is determined at compile time. The availability of other types of database files also depends on compile-time settings.

It is crucial that whatever program you use to create your group files is configured to use the same type of database.

Apache模块 mod_authz_default

说明	在未正确配置授权支持模块的情况下简单拒绝一切授权请求
状态	基本(B)
模块名	authz_default_module
源文件	mod_authz_default.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

该模块是失败补救(fallback)模块，它在未正确配置授权支持模块(比如 `mod_authz_user` 或 `mod_authz_groupfile`)的情况下简单拒绝一切授权请求。

AuthzDefaultAuthoritative 指令

说明	指定是否将授权操作交由更底层的模块来处理
语法	`AuthzDefaultAuthoritative On|Off`
默认值	`AuthzDefaultAuthoritative On`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	基本(B)
模块	mod_authz_default

将 `AuthzDefaultAuthoritative` 明确设置为 `Off` 将允许将授权操作交由更底层的(在 `modules.c` 文件中定义的)模块来处理。

注意

通常并不存在更底层的模块，因为 `mod_authz_default` 已经被定义为非常底层的模块了。因此最好将 `AuthzDefaultAuthoritative` 保持其默认值(`On`)。

Apache模块 mod_authz_groupfile

说明	使用纯文本文件为组提供授权支持
状态	基本(B)
模块名	authz_groupfile_module
源文件	mod_authz_groupfile.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

该模块可以根据经过认证的用户是否属于特定组来允许或拒绝访问受保护的区域。 `mod_authz_dbm` 模块也提供了类似的功能。

AuthGroupFile 指令

说明	设定一个包含用于执行用户认证的用户组列表的纯文本文件
语法	`AuthGroupFile file-path`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	基本(B)
模块	mod_authz_groupfile

`AuthGroupFile` 指令设定一个文本文件的名称，这个纯文本文件包含用于执行用户认证的用户组列表。File-path是存放用户组列表文件的路径。如果不是绝对路径，则是相对于 `ServerRoot` 的相对路径。

这个用户组列表文件每行包含一个用户组名称，后跟一个冒号，再跟该组用户的用户名称，用户名之间以空格分隔。

示例：

```
mygroup: bob joe anne
```

注意：搜索很大的文本文件是非常慢的； `AuthDBMGroupFile` 提供了更出色的性能。

安全

必须确保 `AuthGroupFile` 文件存放在WEB服务器所在目录之外，千万不要放在它所保护的目录中，否则可能会被客户端下载。

AuthzGroupFileAuthoritative 指令

说明	指定是否将授权操作交由更底层的模块来处理
语法	`AuthzGroupFileAuthoritative On|Off`
默认值	`AuthzGroupFileAuthoritative On`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	基本(B)
模块	mod_authz_groupfile

如果将 `AuthzGroupFileAuthoritative` 明确设置为 `Off` ，那么将允许在所提供的userID(用户名)无法匹配任何一个组的情况下，将授权操作交由更底层的(在 `modules.c` 文件中定义的)模块来处理。

默认情况下，不会传递控制权，而是返回一个要求认证的应答。这样做是为了保持系统更加安全并且符合NCSA要求的标准。

安全

务必仔细查看是否允许你的用户在 `.htaccess` 文件中修改这个指令，并确认这的确是你想要的。仅仅看管一个 `.htpasswd` 文件很容易，但是看管一大堆文件却很困难。

Apache模块 mod_authz_host

说明	提供基于主机名、IP地址、请求特征的访问控制
状态	基本(B)
模块名	authz_host_module
源文件	mod_authz_host.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

`mod_authz_host` 提供的指令用在 `<Directory>` , `<Files>` , `<Location>` 段中，也用于 `.htaccess` 文件中控制对服务器特定部分的访问。只要能在环境变量中捕获到主机名、IP地址或其他的客户端请求特征，就可以基于这些特征对访问进行控制。 `Allow` 和 `Deny` 指令用于指出允许哪些客户及不允许哪些客户访问服务器，而 `Order` 指令设置默认的访问状态并配置 `Allow` 和 `Deny` 指令怎样相互作用。

基于主机的访问控制和基于口令的身份验证两套机制可以同时实现。在这种情况下， `Satisfy` 指令用来决定两套机制如何相互作用。

一般来说，访问控制指令适用于所有的访问方法(`GET` , `PUT` , `POST` 等)。在多数情况下这是一个被期望的特性。但是，只限制某些方法而对其他方法不加限制也是可能的：通过把指令放到一个 `<Limit>` 段中即可。

Allow 指令

说明	控制哪些主机能够访问服务器的该区域
语法	`Allow from all|host|env=env-variable [host|env=env-variable] ...`
作用域	directory, .htaccess
覆盖项	Limit
状态	基本(B)
模块	mod_authz_host

`Allow` 指令控制哪些主机可以访问服务器的该区域。可以根据主机名、IP地址、 IP地址范围或其他环境变量中捕获的客户端请求特性进行控制。

这个指令的第一个参数总是" `from` "，随后的参数可以有三种不同形式：如果指定" `Allow from all` "，则允许所有主机访问，按照下述 `Deny` 和 `Order` 指令的配置。若要只允许特定的主机或主机群访问服务器，*host*可以用下面任何一种格式来指定：

一个(部分)域名

示例：

```
Allow from apache.org
Allow from .net example.edu
```

主机名与给定字符串匹配或者以给定字符串结尾的主机允许访问。只有完整的名字组成部分才被匹配，因此上述例子将匹配 `foo.apache.org` 但不能匹配 `fooapache.org` 。这样的配置将导致Apache不管 `HostnameLookups` 指令是如何设置的，对一个对客户IP地址都要执行两次DNS查询：一次正查询保证IP没有伪造，一次反查询保证主机名没有伪造。只有两次查询的结果都吻合，并且主机名能够被匹配，访问才被允许。

完整的IP地址

示例：

```
Allow from 10.1.2.3
Allow from 192.168.1.104 192.168.1.205
```

允许拥有这些IP地址的主机进行访问。

部分IP地址

示例：

```
Allow from 10.1

Allow from 10 172.20 192.168.2
```

IP地址的开始1到3个字节，用于子网限制。

网络/掩码对

示例：

```
Allow from 10.1.0.0/255.255.0.0
```

一个网络"a.b.c.d"和一个掩码"w.x.y.z"，用于更精确的子网限制。

网络/nnn无类别域间路由规格(CIDR specification)

示例：

```
Allow from 10.1.0.0/16
```

同前一种情况相似，除了掩码由nnn个高位字节构成。

注意以上例子中的后三个匹配完全相同的一组主机。

IPv6地址和IPv6子网可以像下面这样指定：

```
Allow from 2001:db8::a00:20ff:fea7:ccea

Allow from 2001:db8::a00:20ff:fea7:ccea/10
```

`Allow` 指令的第三种参数格式允许对服务器的访问由环境变量的一个扩展指定。指定" `Allow from env=env-variable` "时，如果环境变量env-variable存在则访问被允许。使用由 `mod_setenvif` 提供的指令，服务器用一种基于客户端请求的弹性方式提供了设置环境变量的能力。因此，这条指令可以用于允许基于像 `User-Agent` (浏览器类型)、 `Referer` 或其他HTTP请求头字段的访问。

示例：

```
      SetEnvIf User-Agent ^KnockKnock/2\.0 let_me_in

      <Directory /docroot>

 Order Deny,Allow

        Deny from all

        Allow from env=let_me_in
      </Directory>
```

这种情况下，发送以`KnockKnock/2.0`开头的用户代理标示的浏览器将被允许访问，而所有其他浏览器将被禁止访问。

Deny 指令

说明	控制哪些主机被禁止访问服务器
语法	`Deny from all|host|env=env-variable [host|env=env-variable] ...`
作用域	directory, .htaccess
覆盖项	Limit
状态	基本(B)
模块	mod_authz_host

这条指令允许基于主机名、IP地址或者环境变量限制对服务器的访问。`Deny`指令的参数设置和`Allow`指令完全相同。

Order 指令

说明	控制默认的访问状态与 **`Allow`** 和 **`Deny`** 指令生效的顺序
语法	`Order ordering`
默认值	`Order Deny,Allow`
作用域	directory, .htaccess
覆盖项	Limit
状态	基本(B)
模块	mod_authz_host

`Order` 指令控制默认的访问状态与 `Allow` 和 `Deny` 指令生效的顺序。Ordering取值范围是以下几种范例之一：

`Deny,Allow`

`Deny` 指令在 `Allow` 指令之前被评估。默认允许所有访问。任何不匹配 `Deny` 指令或者匹配 `Allow` 指令的客户都被允许访问。

`Allow,Deny`

`Allow` 指令在 `Deny` 指令之前被评估。默认拒绝所有访问。任何不匹配 `Allow` 指令或者匹配 `Deny` 指令的客户都将被禁止访问。

`Mutual-failure`

只有出现在 `Allow` 列表并且不出现在 `Deny` 列表中的主机才被允许访问。这种顺序与" `Order Allow,Deny` "具有同样效果，不赞成使用。

关键字只能用逗号分隔；它们之间不能有空格。注意在所有情况下每个 `Allow` 和 `Deny` 指令语句都将被评估。

在下面的例子中，apache.org域中所有主机都允许访问，而其他任何主机的访问都将被拒绝。

```
Order Deny,Allow
Deny from all
Allow from apache.org
```

下面例子中，apache.org域中所有主机，除了foo.apache.org子域包含的主机被拒绝以外，其他都允许访问。而所有不在apache.org域中的主机都不允许访问，因为默认状态是拒绝对服务器的访问。

```
Order Allow,Deny
Allow from apache.org
Deny from foo.apache.org
```

另一方面，如果上个例子中的 `Order` 指令改变为" `Deny,Allow` "，将允许所有主机的访问。这是因为，不管配置文件中指令的实际顺序如何，" `Allow from apache.org` "指令会最后被评估到并覆盖之前的" `Deny from foo.apache.org` "。所有不在 `apache.org` 域中的主机也允许访问是因为默认状态被改变到了允许。

即使没有伴随 `Allow` 和 `Deny` 指令，一个 `Order` 指令的存在也会影响到服务器上某一个部分的访问，这是由于它对默认访问状态的影响。例如：

```
     <Directory /www>

Order Allow,Deny
     </Directory>
```

这样将会禁止所有对 `/www` 目录的访问，因为默认状态将被设置为拒绝。

`Order` 指令只在服务器配置的每个段内部控制访问指令的处理。这暗示着，例如，一个在 `<Location>` 段中出现的 `Allow` 或 `Deny` 指令总是将会在一个 `<Directory>` 段或者 `.htaccess` 文件中出现的 `Allow` 或 `Deny` 指令之后被评估，而不管 `Order` 指令如何设置。要了解配置段落合并的详细信息，参见配置段文档。

Apache模块 mod_authz_owner

说明	基于文件的所有者进行授权
状态	扩展(E)
模块名	authz_owner_module
源文件	mod_authz_owner.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

This module authorizes access to files by comparing the userid used for HTTP authentication (the web userid) with the file-system owner or group of the requested file. The supplied username and password must be already properly verified by an authentication module, such as `mod_auth_basic` 或 `mod_auth_digest` . `mod_authz_owner` recognizes two arguments for the `Require` directive, `file-owner` 和 `file-group` , as follows:

`file-owner`

The supplied web-username must match the system's name for the owner of the file being requested. That is, if the operating system says the requested file is owned by `jones` , then the username used to access it through the web must be `jones` as well.

`file-group`

The name of the system group that owns the file must be present in a group database, which is provided, for example, by `mod_authz_groupfile` 或 `mod_authz_dbm` , and the web-username must be a member of that group. For example, if the operating system says the requested file is owned by (system) group `accounts` , the group `accounts` must appear in the group database and the web-username used in the request must be a member of that group.

注意

If `mod_authz_owner` is used in order to authorize a resource that is not actually present in the filesystem (*i.e.* a virtual resource), it will deny the access.

Particularly it will never authorize content negotiated "MultiViews" resources.

Configuration Examples

Require file-owner

Consider a multi-user system running the Apache Web server, with each user having his or her own files in `~/public_html/private`. Assuming that there is a single `AuthDBMUserFile` database that lists all of their web-usernames, and that these usernames match the system's usernames that actually own the files on the server, then the following stanza would allow only the user himself access to his own files. User `jones` would not be allowed to access files in `/home/smith/public_html/private` unless they were owned by `jones` instead of `smith`.

```
        <Directory /home/*/public_html/private>

  AuthType Basic

          AuthName MyPrivateFiles

          AuthBasicProvider dbm

          AuthDBMUserFile /usr/local/apache2/etc/.htdbm-all

          Satisfy All

          Require file-owner
        </Directory>
```

Require file-group

Consider a system similar to the one described above, but with some users that share their project files in `~/public_html/project-foo`. The files are owned by the system group `foo` and there is a single `AuthDBMGroupFile` database that contains all of the web-usernames and their group membership, *i.e.* they must be at least member of a group named `foo`. So if `jones` 和 `smith` are both member of the group `foo`, then both will be authorized to access the `project-foo` directories of each other.

```
        <Directory /home/*/public_html/project-foo>

  AuthType Basic

          AuthName "Project Foo Files"

          AuthBasicProvider dbm

          # combined user/group database

          AuthDBMUserFile  /usr/local/apache2/etc/.htdbm-all

          AuthDBMGroupFile /usr/local/apache2/etc/.htdbm-all

          Satisfy All

          Require file-group
        </Directory>
```

AuthzOwnerAuthoritative 指令

说明	Sets whether authorization will be passed on to lower level modules
语法	`AuthzOwnerAuthoritative On|Off`
默认值	`AuthzOwnerAuthoritative On`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_authz_owner

Setting the `AuthzOwnerAuthoritative` directive explicitly to `Off` allows for user authorization to be passed on to lower level modules (as defined in the `modules.c` files) if:

- in the case of `file-owner` the file-system owner does not match the supplied web-username or could not be determined, or
- in the case of `file-group` the file-system group does not contain the supplied web-username or could not be determined.

Note that setting the value to `Off` also allows the combination of `file-owner` 和 `file-group`, so access will be allowed if either one or the other (or both) match.

By default, control is not passed on and an authorization failure will result in an "Authentication Required" reply. Not setting it to `Off` thus keeps the system secure and forces an NCSA compliant behaviour.

Apache模块 mod_authz_user

说明	基于每个用户提供授权支持
状态	基本(B)
模块名	authz_user_module
源文件	mod_authz_user.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

该模块可以允许或拒绝经过认证的用户访问受保护的区域。 `mod_authz_user` 有两种授权方式(二选一)：根据在 `Require user` 中列出的用户对访问进行控制，或者根据 `require valid-user` 指令简单允许所有成功通过认证的用户进行访问。

AuthzUserAuthoritative 指令

说明	指定是否将授权操作交由更底层的模块来处理
语法	`AuthzUserAuthoritative On|Off`
默认值	`AuthzUserAuthoritative On`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	基本(B)
模块	mod_authz_user

如果将 `AuthzUserAuthoritative` 明确设置为 `Off` ，那么将允许在所提供的userID(用户名)无法匹配任何一个用户的情况下，将授权操作交由更底层的(在 `modules.c` 文件中定义的)模块来处理。

默认情况下，不会传递控制权，而是返回一个要求认证的应答。这样做是为了保持系统更加安全并且符合NCSA要求的标准。

Apache模块 mod_autoindex

说明	自动对目录中的内容生成列表，类似于"ls"或"dir"命令
状态	基本(B)
模块名	autoindex_module
源文件	mod_autoindex.c

概述

The index of a directory can come from one of two sources:

- A file written by the user, typically called `index.html`. The `DirectoryIndex` directive sets the name of this file. This is controlled by `mod_dir`.
- Otherwise, a listing generated by the server. The other directives control the format of this listing. The `AddIcon`, `AddIconByEncoding` 和 `AddIconByType` are used to set a list of icons to display for various file types; for each file listed, the first icon listed that matches the file is displayed. These are controlled by `mod_autoindex`.

The two functions are separated so that you can completely remove (or replace) automatic index generation should you want to.

Automatic index generation is enabled with using `Options +Indexes`. See the `Options` directive for more details.

If the `FancyIndexing` option is given with the `IndexOptions` directive, the column headers are links that control the order of the display. If you select a header link, the listing will be regenerated, sorted by the values in that column. Selecting the same header repeatedly toggles between ascending and descending order. These column header links are suppressed with `IndexOptions` directive's `SuppressColumnSorting` option.

Note that when the display is sorted by "Size", it's the *actual* size of the files that's used, not the displayed value - so a 1010-byte file will always be displayed before a 1011-byte file (if in ascending order) even though they both are shown as "1K".

Autoindex Request Query Arguments

Apache 2.0.23 reorganized the Query Arguments for Column Sorting, and introduced an entire group of new query options. To effectively eliminate all client control over the output, the `IndexOptions IgnoreClient` option was introduced.

The column sorting headers themselves are self-referencing hyperlinks that add the sort query options shown below. Any option below may be added to any request for the directory resource.

- `C=N` sorts the directory by file name
- `C=M` sorts the directory by last-modified date, then file name
- `C=S` sorts the directory by size, then file name
- `C=D` sorts the directory by description, then file name
- `O=A` sorts the listing in Ascending Order
- `O=D` sorts the listing in Descending Order
- `F=0` formats the listing as a simple list (not FancyIndexed)
- `F=1` formats the listing as a FancyIndexed list
- `F=2` formats the listing as an HTMLTable FancyIndexed list
- `V=0` disables version sorting
- `V=1` enables version sorting
- `P=pattern` lists only files matching the given pattern

Note that the 'P'attern query argument is tested *after* the usual `IndexIgnore` directives are processed, and all file names are still subjected to the same criteria as any other autoindex listing. The Query Arguments parser in `mod_autoindex` will stop abruptly when an unrecognized option is encountered. The Query Arguments must be well formed, according to the table above.

The simple example below, which can be clipped and saved in a header.html file, illustrates these query options. Note that the unknown "X" argument, for the submit button, is listed last to assure the arguments are all parsed before mod_autoindex encounters the X=Go input.

```
      <form action="" method="get">

  Show me a <select name="F">

  <option value="0"> Plain list</option>

           <option value="1" selected="selected"> Fancy list</option>

           <option value="2"> Table list</option>
         </select>

         Sorted by <select name="C">

  <option value="N" selected="selected"> Name</option>

           <option value="M"> Date Modified</option>

           <option value="S"> Size</option>

           <option value="D"> Description</option>
         </select>

         <select name="O">

  <option value="A" selected="selected"> Ascending</option>

           <option value="D"> Descending</option>
         </select>

         <select name="V">

  <option value="0" selected="selected"> in Normal order</option>

           <option value="1"> in Version order</option>
         </select>

         Matching <input type="text" name="P" value="*" />

         <input type="submit" name="X" value="Go" />
      </form>
```

AddAlt 指令

说明	**Alternate text to display for a file, instead of an icon selected by filename**
语法	`AddAlt string file [file] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

`AddAlt` provides the alternate text to display for a file, instead of an icon, for `FancyIndexing` . File is a file extension, partial filename, wild-card expression or full filename for files to describe. If String contains any whitespace, you have to enclose it in quotes

(`"` 或 `'`). This alternate text is displayed if the client is image-incapable, has image loading disabled, or fails to retrieve the icon.

例子

```
AddAlt "PDF file" *.pdf

AddAlt Compressed *.gz *.zip *.Z
```

AddAltByEncoding 指令

说明	Alternate text to display for a file instead of an icon selected by MIME-encoding
语法	`AddAltByEncoding string MIME-encoding [MIME-encoding] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

`AddAltByEncoding` provides the alternate text to display for a file, instead of an icon, for `FancyIndexing` . MIME-encoding is a valid content-encoding, such as `x-compress` . If String contains any whitespace, you have to enclose it in quotes (`"` 或 `'`). This alternate text is displayed if the client is image-incapable, has image loading disabled, or fails to retrieve the icon.

示例

```
AddAltByEncoding gzip x-gzip
```

AddAltByType 指令

说明	Alternate text to display for a file, instead of an icon selected by MIME content-type
语法	`AddAltByType string MIME-type [MIME-type] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

`AddAltByType` sets the alternate text to display for a file, instead of an icon, for `FancyIndexing`. MIME-type is a valid content-type, such as `text/html`. If String contains any whitespace, you have to enclose it in quotes (`"` 或 `'`). This alternate text is displayed if the client is image-incapable, has image loading disabled, or fails to retrieve the icon.

示例

```
AddAltByType 'plain text' text/plain
```

AddDescription 指令

说明	Description to display for a file
语法	`AddDescription string file [file] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

This sets the description to display for a file, for `FancyIndexing`. File is a file extension, partial filename, wild-card expression or full filename for files to describe. String is enclosed in double quotes (`"`).

示例

```
AddDescription "The planet Mars" /web/pics/mars.gif
```

The typical, default description field is 23 bytes wide. 6 more bytes are added by the `IndexOptions SuppressIcon` option, 7 bytes are added by the `IndexOptions SuppressSize` option, and 19 bytes are added by the `IndexOptions SuppressLastModified` option. Therefore, the widest default the description column is ever assigned is 55 bytes.

See the DescriptionWidth `IndexOptions` keyword for details on overriding the size of this column, or allowing descriptions of unlimited length.

Caution

Descriptive text defined with `AddDescription` may contain HTML markup, such as tags and character entities. If the width of the description column should happen to truncate a tagged element (such as cutting off the end of a bolded phrase), the results may affect the rest of the directory listing.

AddIcon 指令

说明	Icon to display for a file selected by name
语法	`AddIcon icon name [name] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

This sets the icon to display next to a file ending in name for `FancyIndexing`. Icon is either a (%-escaped) relative URL to the icon, or of the format `(alttext, url)` where alttext is the text tag given for an icon for non-graphical browsers.

Name is either `^^DIRECTORY^^` for directories, `^^BLANKICON^^` for blank lines (to format the list correctly), a file extension, a wildcard expression, a partial filename or a complete filename.

例子

```
AddIcon (IMG,/icons/image.xbm) .gif .jpg .xbm
AddIcon /icons/dir.xbm ^^DIRECTORY^^
AddIcon /icons/backup.xbm *~
```

`AddIconByType` should be used in preference to `AddIcon`, when possible.

AddIconByEncoding 指令

说明	Icon to display next to files selected by MIME content-encoding
语法	`AddIconByEncoding icon MIME-encoding [MIME-encoding] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

This sets the icon to display next to files with `FancyIndexing`. Icon is either a (%-escaped) relative URL to the icon, or of the format `(alttext, url)` where alttext is the text tag given for an icon for non-graphical browsers.

MIME-encoding is a wildcard expression matching required the content-encoding.

示例

```
AddIconByEncoding /icons/compress.xbm x-compress
```

AddIconByType 指令

说明	Icon to display next to files selected by MIME content-type
语法	`AddIconByType icon MIME-type [MIME-type] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

This sets the icon to display next to files of type MIME-type for `FancyIndexing`. Icon is either a (%-escaped) relative URL to the icon, or of the format `(alttext, url)` where alttext is the text tag given for an icon for non-graphical browsers.

MIME-type is a wildcard expression matching required the mime types.

示例

```
AddIconByType (IMG,/icons/image.xbm) image/*
```

DefaultIcon 指令

说明	Icon to display for files when no specific icon is configured
语法	`DefaultIcon url-path`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

`DefaultIcon` directive sets the icon to display for files when no specific icon is known, for `FancyIndexing`. Url-path is a (%-escaped) relative URL to the icon.

示例

```
DefaultIcon /icon/unknown.xbm
```

HeaderName 指令

说明	Name of the file that will be inserted at the top of the index listing
语法	`HeaderName filename`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

`HeaderName` directive sets the name of the file that will be inserted at the top of the index listing. Filename is the name of the file to include.

示例

```
HeaderName HEADER.html
```

Both HeaderName and `ReadmeName` now treat Filename as a URI path relative to the one used to access the directory being indexed. If Filename begins with a slash, it will be taken to be relative to the `DocumentRoot`.

示例

```
HeaderName /include/HEADER.html
```

Filename must resolve to a document with a major content type of `text/*` (例如, `text/html` , `text/plain` , etc.). This means that filename may refer to a CGI script if the script's actual file type (as opposed to its output) is marked as `text/html` such as with a directive like:

```
AddType text/html .cgi
```

Content negotiation will be performed if `Options` `MultiViews` is in effect. If filename resolves to a static `text/html` document (not a CGI script) and either one of the `options` `Includes` 或 `IncludesNOEXEC` is enabled, the file will be processed for server-side includes (see the `mod_include` documentation).

If the file specified by `HeaderName` contains the beginnings of an HTML document (<html>, <head>, etc.) then you will probably want to set `IndexOptions +SuppressHTMLPreamble` , so that these tags are not repeated.

IndexIgnore 指令

说明	Adds to the list of files to hide when listing a directory
语法	`IndexIgnore file [file] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

`IndexIgnore` directive adds to the list of files to hide when listing a directory. File is a shell-style wildcard expression or full filename. Multiple IndexIgnore directives add to the list, rather than the replacing the list of ignored files. By default, the list contains `.` (the current directory).

```
IndexIgnore README .htaccess *.bak *~
```

IndexOptions 指令

说明	Various configuration settings for directory indexing
语法	`IndexOptions [+|-]option [[+|-]option] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

`IndexOptions` directive specifies the behavior of the directory indexing. Option can be one of

DescriptionWidth=[n | *] (*Apache 2.0.23 and later*)

The `DescriptionWidth` keyword allows you to specify the width of the description column in characters.

`-DescriptionWidth` (or unset) allows `mod_autoindex` to calculate the best width.

`DescriptionWidth=n` fixes the column width to n bytes wide.

`DescriptionWidth=*` grows the column to the width necessary to accommodate the longest description string.

See the section on `AddDescription` for dangers inherent in truncating descriptions.

FancyIndexing

This turns on fancy indexing of directories.

FoldersFirst (*Apache 2.0.23 and later*)

If this option is enabled, subdirectory listings will *always* appear first, followed by normal files in the directory. The listing is basically broken into two components, the files and the subdirectories, and each is sorted separately and then displayed subdirectories-first. For instance, if the sort order is descending by name, and `FoldersFirst` is enabled, subdirectory `Zed` will be listed before subdirectory `Beta` , which will be listed before normal files `Gamma` 和 `Alpha` . **This option only has an effect if `FancyIndexing` is also enabled.**

HTMLTable (*Experimental, Apache 2.0.23 and later*)

This experimental option with FancyIndexing constructs a simple table for the fancy directory listing. Note this will confuse older browsers. It is particularly necessary if file names or description text will alternate between left-to-right and right-to-left reading order, as can happen on WinNT or other utf-8 enabled platforms.

IconsAreLinks

This makes the icons part of the anchor for the filename, for fancy indexing.

IconHeight[=pixels]

Presence of this option, when used with IconWidth, will cause the server to include `height` 和 `width` attributes in the `img` tag for the file icon. This allows browser to precalculate the page layout without having to wait until all the images have been loaded. If no value is given for the option, it defaults to the standard height of the icons supplied with the Apache software.

IconWidth[=pixels]

Presence of this option, when used with `IconHeight`, will cause the server to include `height` 和 `width` attributes in the `img` tag for the file icon. This allows browser to precalculate the page layout without having to wait until all the images have been loaded. If no value is given for the option, it defaults to the standard width of the icons supplied with the Apache software.

IgnoreCase

If this option is enabled, names are sorted in a case-insensitive manner. For instance, if the sort order is ascending by name, and IgnoreCase is enabled, file Zeta will be listed after file alfa (Note: file GAMMA will always be listed before file gamma).

IgnoreClient

This option causes `mod_autoindex` to ignore all query variables from the client, including sort order (implies `SuppressColumnSorting`.)

NameWidth=[n | *]

The `NameWidth` keyword allows you to specify the width of the filename column in bytes.

`-NameWidth` (or unset) allows `mod_autoindex` to calculate the best width.

`NameWidth=n` fixes the column width to n bytes wide.

`NameWidth=*` grows the column to the necessary width.

ScanHTMLTitles

This enables the extraction of the title from HTML documents for fancy indexing. If the file does not have a description given by `AddDescription` then httpd will read the document for the value of the `title` element. This is CPU and disk intensive.

ShowForbidden

If specified, Apache will show files normally hidden because the subrequest returned HTTP_UNAUTHORIZED or HTTP_FORBIDDEN

SuppressColumnSorting

If specified, Apache will not make the column headings in a FancyIndexed directory listing into links for sorting. The default behavior is for them to be links; selecting the column heading will sort the directory listing by the values in that column. **Prior to Apache 2.0.23, this also disabled parsing the Query Arguments for the sort string.** That behavior is now controlled by IndexOptions IgnoreClient in Apache 2.0.23.

SuppressDescription

This will suppress the file description in fancy indexing listings. By default, no file descriptions are defined, and so the use of this option will regain 23 characters of screen space to use for something else. See `AddDescription` for information about setting the file description. See also the `DescriptionWidth` index option to limit the size of the description column.

SuppressHTMLPreamble

If the directory actually contains a file specified by the `HeaderName` directive, the module usually includes the contents of the file after a standard HTML preamble (`<html>` , `<head>` , *et cetera*). The `SuppressHTMLPreamble` option disables this behaviour, causing the module to start the display with the header file contents. The header file must contain appropriate HTML instructions in this case. If there is no header file, the preamble is generated as usual.

SuppressIcon (*Apache 2.0.23 and later*)

This will suppress the icon in fancy indexing listings. Combining both `SuppressIcon` 和 `SuppressRules` yields proper HTML 3.2 output, which by the final specification prohibits `img` 和 `hr` elements from the `pre` block (used to format FancyIndexed listings.)

SuppressLastModified

This will suppress the display of the last modification date, in fancy indexing listings.

SuppressRules (*Apache 2.0.23 and later*)

This will suppress the horizontal rule lines (`hr` elements) in directory listings. Combining both `SuppressIcon` 和 `SuppressRules` yields proper HTML 3.2 output, which by the final specification prohibits `img` 和 `hr` elements from the `pre` block (used to format FancyIndexed listings.)

SuppressSize

This will suppress the file size in fancy indexing listings.

TrackModified (*Apache 2.0.23 and later*)

This returns the Last-Modified and ETag values for the listed directory in the HTTP header. It is only valid if the operating system and file system return appropriate stat() results. Some Unix systems do so, as do OS2's JFS and Win32's NTFS volumes. OS2 and Win32 FAT volumes, for example, do not. Once this feature is enabled, the client or proxy can track changes to the list of files when they perform a `HEAD` request. Note some operating systems correctly track new and removed files, but do not track changes for sizes or dates of the files within the directory. **Changes to the size or date stamp of an existing file will not update the Last-Modified header on all Unix platforms.** If this is a concern, leave this option disabled.

VersionSort (*Apache 2.0a3 and later*)

The `VersionSort` keyword causes files containing version numbers to sort in a natural way. Strings are sorted as usual, except that substrings of digits in the name and description are compared according to their numeric value.

示例：

```
foo-1.7
foo-1.7.2
foo-1.7.12
foo-1.8.2
foo-1.8.2a
foo-1.12
```

If the number starts with a zero, then it is considered to be a fraction:

```
foo-1.001
foo-1.002
foo-1.030
foo-1.04
```

XHTML (*Apache 2.0.49 and later*)

The `XHTML` keyword forces `mod_autoindex` to emit XHTML 1.0 code instead of HTML 3.2.

Incremental IndexOptions

Apache 1.3.3 introduced some significant changes in the handling of `IndexOptions` directives. In particular:

- Multiple `IndexOptions` directives for a single directory are now merged together. The result of:

  ```
  &lt;Directory /foo&gt;
  IndexOptions HTMLTable

    IndexOptions SuppressColumnsorting
  &lt;/Directory&gt;
  ```

  will be the equivalent of

  ```
  IndexOptions HTMLTable SuppressColumnsorting
  ```

- The addition of the incremental syntax (*i.e.*, prefixing keywords with `+` 或 `-`).

Whenever a '+' or '-' prefixed keyword is encountered, it is applied to the current `IndexOptions` settings (which may have been inherited from an upper-level directory). However, whenever an unprefixed keyword is processed, it clears all inherited options and any incremental settings encountered so far. Consider the following example:

```
IndexOptions +ScanHTMLTitles -IconsAreLinks FancyIndexing

IndexOptions +SuppressSize
```

The net effect is equivalent to `IndexOptions FancyIndexing +SuppressSize` , because the unprefixed `FancyIndexing` discarded the incremental keywords before it, but allowed them to start accumulating again afterward.

To unconditionally set the `IndexOptions` for a particular directory, clearing the inherited settings, specify keywords without any `+` 或 `-` prefixes.

IndexOrderDefault 指令

说明	**Sets the default ordering of the directory index**
语法	`IndexOrderDefault Ascending|Descending Name|Date|Size|Descripti`
默认值	`IndexOrderDefault Ascending Name`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

`IndexOrderDefault` directive is used in combination with the `FancyIndexing` index option. By default, fancyindexed directory listings are displayed in ascending order by filename; the `IndexOrderDefault` allows you to change this initial display order.

`IndexOrderDefault` takes two arguments. The first must be either `Ascending` 或 `Descending` , indicating the direction of the sort. The second argument must be one of the keywords `Name` , `Date` , `Size` , or `Description` , and identifies the primary key. The secondary key is *always* the ascending filename.

You can force a directory listing to only be displayed in a particular order by combining this directive with the `SuppressColumnSorting` index option; this will prevent the client from requesting the directory listing in a different order.

IndexStyleSheet 指令

说明	**Adds a CSS stylesheet to the directory index**
语法	`IndexStyleSheet url-path`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

`IndexStyleSheet` directive sets the name of the file that will be used as the CSS for the index listing.

示例

```
IndexStyleSheet "/css/style.css"
```

ReadmeName 指令

说明	Name of the file that will be inserted at the end of the index listing
语法	`ReadmeName filename`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_autoindex

`ReadmeName` directive sets the name of the file that will be appended to the end of the index listing. Filename is the name of the file to include, and is taken to be relative to the location being indexed. If Filename begins with a slash, it will be taken to be relative to the `DocumentRoot` .

示例

```
ReadmeName FOOTER.html
```

Example 2

```
ReadmeName /include/FOOTER.html
```

See also `HeaderName` , where this behavior is described in greater detail.

Apache模块 mod_cache

说明	基于URI键的内容动态缓冲(内存或磁盘)
状态	扩展(E)
模块名	cache_module
源文件	mod_cache.c

概述

This module should be used with care and can be used to circumvent `Allow` 和 `Deny` directives. You should not enable caching for any content to which you wish to limit access by client host name, address or environment variable.

`mod_cache` implements an RFC 2616 compliant HTTP content cache that can be used to cache either local or proxied content. `mod_cache` requires the services of one or more storage management modules. Two storage management modules are included in the base Apache distribution:

`mod_disk_cache`

implements a disk based storage manager.

`mod_mem_cache`

implements a memory based storage manager. `mod_mem_cache` can be configured to operate in two modes: caching open file descriptors or caching objects in heap storage. `mod_mem_cache` can be used to cache locally generated content or to cache backend server content for `mod_proxy` when configured using `ProxyPass` (aka <dfn class="calibre27">reverse proxy</dfn>)

Content is stored in and retrieved from the cache using URI based keys. Content with access protection is not cached.

Related Modules and Directives

相关模块

- `mod_disk_cache`
- `mod_mem_cache`

相关指令

- `CacheRoot`
- `CacheSize`
- `CacheDirLevels`
- `CacheDirLength`
- `CacheMinFileSize`
- `CacheMaxFileSize`
- `MCacheSize`
- `MCacheMaxObjectCount`
- `MCacheMinObjectSize`
- `MCacheMaxObjectSize`
- `MCacheRemovalAlgorithm`
- `MCacheMaxStreamingBuffer`

Sample Configuration

Sample httpd.conf

```
    #
    # Sample Cache Configuration
    #
    LoadModule cache_module modules/mod_cache.so
    <IfModule mod_cache.c>
#LoadModule disk_cache_module modules/mod_disk_cache.so
      # If you want to use mod_disk_cache instead of mod_mem_cache,
      # uncomment the line above and comment out the LoadModule line below.
      <IfModule mod_disk_cache.c>
CacheRoot c:/cacheroot
        CacheEnable disk  /
        CacheDirLevels 5
        CacheDirLength 3
      </IfModule>
      LoadModule mem_cache_module modules/mod_mem_cache.so
      <IfModule mod_mem_cache.c>
CacheEnable mem  /
        MCacheSize 4096
        MCacheMaxObjectCount 100
        MCacheMinObjectSize 1
        MCacheMaxObjectSize 2048
      </IfModule>
      # When acting as a proxy, don't cache the list of security updates
      CacheDisable http://security.update.server/update-list/
    </IfModule>
```

CacheDefaultExpire 指令

说明	**The default duration to cache a document when no expiry date is specified.**
语法	`CacheDefaultExpire seconds`
默认值	`CacheDefaultExpire 3600 (one hour)`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_cache

`CacheDefaultExpire` directive specifies a default time, in seconds, to cache a document if neither an expiry date nor last-modified date are provided with the document. The value specified with the `CacheMaxExpire` directive does *not* override this setting.

```
CacheDefaultExpire 86400
```

CacheDisable 指令

说明	Disable caching of specified URLs
语法	`CacheDisable url-string`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_cache

`CacheDisable` directive instructs `mod_cache` to *not* cache urls at or below url-string.

示例

```
CacheDisable /local_files
```

CacheEnable 指令

说明	Enable caching of specified URLs using a specified storage manager
语法	`CacheEnable cache_type url-string`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_cache

`CacheEnable` directive instructs `mod_cache` to cache urls at or below url-string. The cache storage manager is specified with the cache_type argument. cache_type `mem` instructs `mod_cache` to use the memory based storage manager implemented by `mod_mem_cache`. cache_type `disk` instructs `mod_cache` to use the disk based storage manager implemented by `mod_disk_cache`. cache_type `fd` instructs `mod_cache` to use the file descriptor cache implemented by `mod_mem_cache`.

In the event that the URL space overlaps between different `CacheEnable` directives (as in the example below), each possible storage manager will be run until the first one that actually processes the request. The order in which the storage managers are run is determined by the order of the `CacheEnable` directives in the configuration file.

```
CacheEnable  mem   /manual
CacheEnable  fd    /images
CacheEnable  disk  /
```

When acting as a forward proxy server, url-string can also be used to specify remote sites and proxy protocols which caching should be enabled for.

```
# Cache proxied url's
CacheEnable  disk  /
# Cache FTP-proxied url's
CacheEnable  disk  ftp://
# Cache content from www.apache.org
CacheEnable  disk  http://www.apache.org/
```

CacheIgnoreCacheControl 指令

说明	Ignore request to not serve cached content to client
语法	`CacheIgnoreCacheControl On|Off`
默认值	`CacheIgnoreCacheControl Off`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_cache

Ordinarily, requests containing a Cache-Control: no-cache or Pragma: no-cache header value will not be served from the cache. The `CacheIgnoreCacheControl` directive allows this behavior to be overridden. `CacheIgnoreCacheControl` On tells the server to attempt to serve the resource from the cache even if the request contains no-cache header values. Resources requiring authorization will *never* be cached.

```
CacheIgnoreCacheControl On
```

Warning:

This directive will allow serving from the cache even if the client has requested that the document not be served from the cache. This might result in stale content being served.

参见

- `CacheStorePrivate`
- `CacheStoreNoStore`

CacheIgnoreHeaders 指令

说明	Do not store the given HTTP header(s) in the cache.
语法	`CacheIgnoreHeaders header-string [header-string] ...`
默认值	`CacheIgnoreHeaders None`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_cache

According to RFC 2616, hop-by-hop HTTP headers are not stored in the cache. The following HTTP headers are hop-by-hop headers and thus do not get stored in the cache in *any* case regardless of the setting of `CacheIgnoreHeaders` :

- `Connection`
- `Keep-Alive`
- `Proxy-Authenticate`
- `Proxy-Authorization`
- `TE`
- `Trailers`
- `Transfer-Encoding`
- `Upgrade`

`CacheIgnoreHeaders` specifies additional HTTP headers that should not to be stored in the cache. For example, it makes sense in some cases to prevent cookies from being stored in the cache.

`CacheIgnoreHeaders` takes a space separated list of HTTP headers that should not be stored in the cache. If only hop-by-hop headers not should be stored in the cache (the RFC 2616 compliant behaviour), `CacheIgnoreHeaders` can be set to `None` .

Example 1

```
CacheIgnoreHeaders Set-Cookie
```

Example 2

```
CacheIgnoreHeaders None
```

Warning:

If headers like `Expires` which are needed for proper cache management are not stored due to a `CacheIgnoreHeaders` setting, the behaviour of mod_cache is undefined.

CacheIgnoreNoLastMod 指令

说明	Ignore the fact that a response has no Last Modified header.
语法	`CacheIgnoreNoLastMod On|Off`
默认值	`CacheIgnoreNoLastMod Off`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_cache

Ordinarily, documents without a last-modified date are not cached. Under some circumstances the last-modified date is removed (during `mod_include` processing for example) or not provided at all. The `CacheIgnoreNoLastMod` directive provides a way to specify that documents without last-modified dates should be considered for caching, even without a last-modified date. If neither a last-modified date nor an expiry date are provided with the document then the value specified by the `CacheDefaultExpire` directive will be used to generate an expiration date.

```
CacheIgnoreNoLastMod On
```

CacheLastModifiedFactor 指令

说明	The factor used to compute an expiry date based on the LastModified date.
语法	`CacheLastModifiedFactor float`
默认值	`CacheLastModifiedFactor 0.1`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_cache

In the event that a document does not provide an expiry date but does provide a last-modified date, an expiry date can be calculated based on the time since the document was last modified. The `CacheLastModifiedFactor` directive specifies a factor to be used in the generation of this expiry date according to the following formula:
`expiry-period = time-since-last-modified-date * factor expiry-date = current-date + expiry`
For example, if the document was last modified 10 hours ago, and factor is 0.1 then the expiry-period will be set to 10*0.1 = 1 hour. If the current time was 3:00pm then the computed expiry-date would be 3:00pm + 1hour = 4:00pm. If the expiry-period would be longer than that set by `CacheMaxExpire`, then the latter takes precedence.

```
CacheLastModifiedFactor 0.5
```

CacheMaxExpire 指令

说明	The maximum time in seconds to cache a document
语法	`CacheMaxExpire seconds`
默认值	`CacheMaxExpire 86400 (one day)`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_cache

`CacheMaxExpire` directive specifies the maximum number of seconds for which cachable HTTP documents will be retained without checking the origin server. Thus, documents will be out of date at most this number of seconds. This maximum value is enforced even if an expiry date was supplied with the document.

```
CacheMaxExpire 604800
```

CacheStoreNoStore 指令

说明	**Attempt to cache requests or responses that have been marked as no-store.**
语法	`CacheStoreNoStore On|Off`
默认值	`CacheStoreNoStore Off`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_cache

Ordinarily, requests or responses with Cache-Control: no-store header values will not be stored in the cache. The `CacheStoreNoCache` directive allows this behavior to be overridden. `CacheStoreNoCache` On tells the server to attempt to cache the resource even if it contains no-store header values. Resources requiring authorization will *never* be cached.

```
CacheStoreNoStore On
```

Warning:

As described in RFC 2616, the no-store directive is intended to "prevent the inadvertent release or retention of sensitive information (for example, on backup tapes)." Enabling this option could store sensitive information in the cache. You are hereby warned.

参见

- `CacheIgnoreCacheControl`
- `CacheStorePrivate`

CacheStorePrivate 指令

说明	Attempt to cache responses that the server has marked as private
语法	`CacheStorePrivate On|Off`
默认值	`CacheStorePrivate Off`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_cache

Ordinarily, responses with Cache-Control: private header values will not be stored in the cache. The `CacheStorePrivate` directive allows this behavior to be overridden. `CacheStorePrivate` On tells the server to attempt to cache the resource even if it contains private header values. Resources requiring authorization will *never* be cached.

```
CacheStorePrivate On
```

Warning:

This directive will allow caching even if the upstream server has requested that the resource not be cached. This directive is only ideal for a 'private' cache.

参见

- `CacheIgnoreCacheControl`
- `CacheStoreNoStore`

Apache模块 mod_cern_meta

说明	允许Apache使用CERN httpd元文件，从而可以在发送文件时对头进行修改
状态	扩展(E)
模块名	cern_meta_module
源文件	mod_cern_meta.c

概述

Emulate the CERN HTTPD Meta file semantics. Meta files are HTTP headers that can be output in addition to the normal range of headers for each file accessed. They appear rather like the Apache .asis files, and are able to provide a crude way of influencing the Expires: header, as well as providing other curiosities. There are many ways to manage meta information, this one was chosen because there is already a large number of CERN users who can exploit this module.

More information on the CERN metafile semantics is available.

MetaDir 指令

说明	Name of the directory to find CERN-style meta information files
语法	`MetaDir directory`
默认值	`MetaDir .web`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	扩展(E)
模块	mod_cern_meta

Specifies the name of the directory in which Apache can find meta information files. The directory is usually a 'hidden' subdirectory of the directory that contains the file being accessed. Set to " `.` " to look in the same directory as the file:

```
MetaDir .
```

Or, to set it to a subdirectory of the directory containing the files:

```
MetaDir .meta
```

MetaFiles 指令

说明	Activates CERN meta-file processing
语法	`MetaFiles on|off`
默认值	`MetaFiles off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	扩展(E)
模块	mod_cern_meta

Turns on/off Meta file processing on a per-directory basis.

MetaSuffix 指令

说明	File name suffix for the file containg CERN-style meta information
语法	`MetaSuffix suffix`
默认值	`MetaSuffix .meta`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	扩展(E)
模块	mod_cern_meta

Specifies the file name suffix for the file containing the meta information. For example, the default values for the two directives will cause a request to `DOCUMENT_ROOT/somedir/index.html` to look in `DOCUMENT_ROOT/somedir/.web/index.html.meta` and will use its contents to generate additional MIME header information.

示例：

```
MetaSuffix .meta
```

Apache模块 mod_cgi

说明	在非线程型MPM(`prefork`)上提供对CGI脚本执行的支持
状态	基本(B)
模块名	cgi_module
源文件	mod_cgi.c

概述

任何具有MIME类型 `application/x-httpd-cgi` 或者被 `cgi-script` 处理器处理的文件都将被作为CGI脚本对待并由服务器运行，它的输出将被返回给客户端。可以通过两种途径使文件成为CGI脚本，一种是文件具有已由 `AddType` 指令定义的扩展名，另一种是文件位于 `ScriptAlias` 目录中。

当服务器调用一个CGI脚本时会往运行环境中添加一个叫做 `DOCUMENT_ROOT` 的变量。这个变量将包含 `DocumentRoot` 指令的值。

要得到对Apache中使用CGI脚本的介绍，参看使用CGI产生动态内容。

在UNIX下使用多线程型的MPM时，应该使用模块 `mod_cgid` 代替本模块。在用户层面，这两个模块本质上是相同的。

CGI环境变量

Apache将按照CGI规范设置环境变量，有以下几个：

PATH_INFO

如果显式设置了 `AcceptPathInfo` 指令为 `off` ，这个变量将不可用。如果没有明确设置 `AcceptPathInfo` ，默认的行为是 `mod_cgi` 将会接受路径信息(URI里脚本文件名之后的 `/more/path/info`)，即使服务器核心对请求的附加路径信息返回"404 NOT FOUND"错误。省略 `AcceptPathInfo` 指令与将它设置为 `On` 对 `mod_cgi` 请求具有相同的效果。

REMOTE_HOST

这个变量只有在 `HostnameLookups` 指令被设置为" `on` "(默认是"off")并且对访问主机的DNS反查确实找到了主机名时才被设置。

REMOTE_IDENT

这个变量只有在 `IdentityCheck` 被设置为 `on` 并且访问主机支持识别协议时才被设置。注意这个变量不能信赖，因为它很容易被假冒。而且如果客户端与服务器之间存在代理的话，这个变量通常完全没有用处。

REMOTE_USER

这个变量只有当CGI脚本需要身份验证时才被设置。

CGI脚本的调试

调试CGI脚本一直以来都很困难，主要是因为在脚本不能正确运行时不可能研究它的输出(stdout和stderr)。这些指令为发生错误时提供了更详细的错误日志。

CGI日志文件格式

配置好以后，CGI错误日志会记录任何没有正确运行的CGI。每个运行失败的CGI脚本都有几行信息被记录。头两行总是这样的格式：

```
        %% [<var class="calibre40">time</var>] <var class="calibre40">request-line</var>
        %% <var class="calibre40">HTTP-status</var> <var class="calibre40">CGI-script-fil
```

如果错误是CGI脚本无法执行，日志文件会包含以下额外的两行：

```
        %%error
<var class="calibre40">error-message</var>
```

或者，如果错误是脚本执行结果返回了不正确的头信息(经常是由于脚本内部的bug)，会记录以下两行：

```
        %request
<var class="calibre40">All HTTP request headers received</var>
<var class="calibre40">POST or PUT entity (if any)</var>
        %response
<var class="calibre40">All headers output by the CGI script</var>
        %stdout
<var class="calibre40">CGI standard output</var>
        %stderr
<var class="calibre40">CGI standard error</var>
```

如果脚本没有在stdout和stderr上输出可能会没有%stdout和%stderr部分。

ScriptLog 指令

说明	CGI脚本错误日志文件的位置
语法	`ScriptLog file-path`
作用域	server config, virtual host
状态	基本(B)
模块	`mod_cgi` , `mod_cgid`

`ScriptLog` 指令设置了CGI脚本错误日志文件的位置。如果没有设置 `ScriptLog` ，就不会创建错误日志。如果设置了，所有CGI错误都会被记入作为指令参数的文件中。如果是一个相对路径则以 `ServerRoot` 为参照。

示例

```
ScriptLog logs/cgi_log
```

此日志会以运行子进程的用户身份打开，比如由主服务器配置部分的 `User` 指令指定的用户。这意味着，或者该用户对脚本日志所在目录具有写权限，或者日志文件由此用户手工创建并设置为可写。如果你把脚本日志放在你的主日志目录中，不要为了让运行子进程的用户可以写日志而改变目录的权限。

注意，脚本日志是为了给创建CGI脚本提供一个调试特性，而不是要在运行服务时持续保持活动状态。它没有为速度或是效率作优化，而且与专门设计的那些特性不同，在某种程度上它会存在安全问题。

ScriptLogBuffer 指令

说明	记入日志文件的**PUT**或**POST**请求头的最大数量
语法	`ScriptLogBuffer bytes`
默认值	`ScriptLogBuffer 1024`
作用域	server config, virtual host
状态	基本(B)
模块	`mod_cgi` , `mod_cgid`

限制记入日志文件的PUT或者POST内容的大小，防止如果接收到很大内容时日志文件的尺寸增加得太快太大。默认地，最多纪录1024字节，但这个数字可以用此指令改变。

ScriptLogLength 指令

说明	日志文件的大小限制(字节)
语法	`ScriptLogLength bytes`
默认值	`ScriptLogLength 10385760`
作用域	server config, virtual host
状态	基本(B)
模块	`mod_cgi` , `mod_cgid`

`ScriptLogLength` 指令可以用于限制CGI脚本文件的大小。由于日志文件对每个CGI错误纪录(所有请求头、所有脚本输出)许多信息，它有可能会变成一个很大的文件。为了防止无限制的增长引起的问题，这个指令可以用来给CGI日志文件的大小设置一个上限。如果文件大小达到了这个限制，就不会再有信息被写入日志。

Apache模块 mod_cgid

说明	在线程型MPM(`worker`)上用一个外部CGI守护进程执行CGI脚本
状态	基本(B)
模块名	cgid_module
源文件	mod_cgid.c
兼容性	仅用于Unix系统上的线程型MPM

概述

除了优化和下面额外的 `ScriptSock` 指令， `mod_cgid` 表现得与 `mod_cgi` 非常相似。参见 **`mod_cgi`** 以了解关于Apache和CGI的信息细节。

在特定的unix操作系统上，从一个多线程服务器fork一个进程是非常昂贵的操作，因为新进程会复制其父进程的所有线程。为了避免每个CGI调用都导致产生这样的开销， `mod_cgid` 创建一个外部守护进程来负责fork子进程以运行CGI脚本。主服务器使用unix domain套接字与这个守护进程通信。

只要编译时选择了多线程型的MPM支持，这个模块就会默认代替 `mod_cgi` 。在用户层面，此模块在配置和操作上与 `mod_cgi` 是一样的。唯一的例外是有一个额外的指令 `ScriptSock` 给出了用于与cgi守护进程通信的套接字文件名前缀。

ScriptSock 指令

说明	用来与CGI守护进程通信的套接字文件名前缀
语法	`ScriptSock file-path`
默认值	`ScriptSock logs/cgisock`
作用域	server config, virtual host
状态	基本(B)
模块	mod_cgid

此指令设置用来与CGI守护进程通信的套接字文件名前缀(其后附加父进程PID组成完整的文件名)。这个套接字将会用启动Apache服务器的父进程用户权限(通常是root)打开。为了维护与CGI脚本通讯的安全性，不允许其他用户拥有写入套接字所在目录的权限是很重要的。

示例

```
ScriptSock /var/run/cgid.sock
```

Apache模块 mod_charset_lite

说明	允许对页面进行字符集转换
状态	实验(X)
模块名	charset_lite_module
源文件	mod_charset_lite.c

概述

This is an **experimental** module and should be used with care. Experiment with your `mod_charset_lite` configuration to ensure that it performs the desired function.

`mod_charset_lite` allows the administrator to specify the source character set of objects as well as the character set they should be translated into before sending to the client. `mod_charset_lite` does not translate the data itself but instead tells Apache what translation to perform. `mod_charset_lite` is applicable to EBCDIC and ASCII host environments. In an EBCDIC environment, Apache normally translates text content from the code page of the Apache process locale to ISO-8859-1. `mod_charset_lite` can be used to specify that a different translation is to be performed. In an ASCII environment, Apache normally performs no translation, so `mod_charset_lite` is needed in order for any translation to take place.

This module provides a small subset of configuration mechanisms implemented by Russian Apache and its associated `mod_charset`.

Common Problems

Invalid character set names

The character set name parameters of `CharsetSourceEnc` 和 `CharsetDefault` must be acceptable to the translation mechanism used by APR on the system where `mod_charset_lite` is deployed. These character set names are not standardized and are usually not the same as the corresponding values used in http headers. Currently, APR can only use iconv(3), so you can easily test your character set names using the iconv(1) program, as follows:

```
iconv -f charsetsourceenc-value -t charsetdefault-value
```

Mismatch between character set of content and translation rules

If the translation rules don't make sense for the content, translation can fail in various ways, including:

- The translation mechanism may return a bad return code, and the connection will be aborted.
- The translation mechanism may silently place special characters (e.g., question marks) in the output buffer when it cannot translate the input buffer.

CharsetDefault 指令

说明	Charset to translate into
语法	`CharsetDefault charset`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	实验(X)
模块	mod_charset_lite

`CharsetDefault` directive specifies the charset that content in the associated container should be translated to.

The value of the charset argument must be accepted as a valid character set name by the character set support in APR. Generally, this means that it must be supported by iconv.

示例

```
     <Directory /export/home/trawick/apacheinst/htdocs/convert>

CharsetSourceEnc  UTF-16BE

       CharsetDefault    ISO-8859-1
     </Directory>
```

CharsetOptions 指令

说明	Configures charset translation behavior
语法	`CharsetOptions option [option] ...`
默认值	`CharsetOptions DebugLevel=0 NoImplicitAdd`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	实验(X)
模块	mod_charset_lite

`CharsetOptions` directive configures certain behaviors of `mod_charset_lite`. Option can be one of

`DebugLevel=n`

The `DebugLevel` keyword allows you to specify the level of debug messages generated by `mod_charset_lite`. By default, no messages are generated. This is equivalent to `DebugLevel=0`. With higher numbers, more debug messages are generated, and server performance will be degraded. The actual meanings of the numeric values are described with the definitions of the DBGLVL_ constants near the beginning of `mod_charset_lite.c`.

`ImplicitAdd | NoImplicitAdd`

The `ImplicitAdd` keyword specifies that `mod_charset_lite` should implicitly insert its filter when the configuration specifies that the character set of content should be translated. If the filter chain is explicitly configured using the `AddOutputFilter` directive, `NoImplicitAdd` should be specified so that `mod_charset_lite` doesn't add its filter.

CharsetSourceEnc 指令

说明	Source charset of files
语法	`CharsetSourceEnc charset`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	实验(X)
模块	mod_charset_lite

`CharsetSourceEnc` directive specifies the source charset of files in the associated container.

The value of the charset argument must be accepted as a valid character set name by the character set support in APR. Generally, this means that it must be supported by iconv.

示例

```
      <Directory /export/home/trawick/apacheinst/htdocs/convert>

  CharsetSourceEnc  UTF-16BE

        CharsetDefault    ISO-8859-1
      </Directory>
```

The character set names in this example work with the iconv translation support in Solaris 8.

Apache模块 mod_dav

说明	允许Apache提供DAV协议支持
状态	扩展(E)
模块名	dav_module
源文件	mod_dav.c

概述

This module provides class 1 and class 2 WebDAV ('Web-based Distributed Authoring and Versioning') functionality for Apache. This extension to the HTTP protocol allows creating, moving, copying, and deleting resources and collections on a remote web server.

Enabling WebDAV

To enable `mod_dav` , add the following to a container in your `httpd.conf` file:

```
Dav On
```

This enables the DAV file system provider, which is implemented by the `mod_dav_fs` module. Therefore, that module must be compiled into the server or loaded at runtime using the `LoadModule` directive.

In addition, a location for the DAV lock database must be specified in the global section of your `httpd.conf` file using the `DavLockDB` directive:

```
      DavLockDB /usr/local/apache2/var/DavLock
```

The directory containing the lock database file must be writable by the `User` 和 `Group` under which Apache is running.

You may wish to add a `<Limit>` clause inside the `<Location>` directive to limit access to DAV-enabled locations. If you want to set the maximum amount of bytes that a DAV client can send at one request, you have to use the `LimitXMLRequestBody` directive. The "normal" `LimitRequestBody` directive has no effect on DAV requests.

Full Example

```
      DavLockDB /usr/local/apache2/var/DavLock

       <Location /foo>

Dav On

         AuthType Basic

         AuthName DAV

         AuthUserFile user.passwd

         <LimitExcept GET OPTIONS>

require user admin
         </LimitExcept>
       </Location>
```

`mod_dav` is a descendent of Greg Stein's mod_dav for Apache 1.3. More information about the module is available from that site.

安全问题

Since DAV access methods allow remote clients to manipulate files on the server, you must take particular care to assure that your server is secure before enabling `mod_dav`.

Any location on the server where DAV is enabled should be protected by authentication. The use of HTTP Basic Authentication is not recommended. You should use at least HTTP Digest Authentication, which is provided by the `mod_auth_digest` module. Nearly all WebDAV clients support this authentication method. An alternative is Basic Authentication over an SSL enabled connection.

In order for `mod_dav` to manage files, it must be able to write to the directories and files under its control using the `User` 和 `Group` under which Apache is running. New files created will also be owned by this `User` 和 `Group`. For this reason, it is important to control access to this account. The DAV repository is considered private to Apache; modifying files outside of Apache (for example using FTP or filesystem-level tools) should not be allowed.

`mod_dav` may be subject to various kinds of denial-of-service attacks. The `LimitXMLRequestBody` directive can be used to limit the amount of memory consumed in parsing large DAV requests. The `DavDepthInfinity` directive can be used to prevent `PROPFIND` requests on a very large repository from consuming large amounts of memory. Another possible denial-of-service attack involves a client simply filling up all available disk space with many large files. There is no direct way to prevent this in Apache, so you should avoid giving DAV access to untrusted users.

Complex Configurations

One common request is to use `mod_dav` to manipulate dynamic files (PHP scripts, CGI scripts, etc). This is difficult because a `GET` request will always run the script, rather than downloading its contents. One way to avoid this is to map two different URLs to the content, one of which will run the script, and one of which will allow it to be downloaded and manipulated with DAV.

```
Alias /phparea /home/gstein/php_files

Alias /php-source /home/gstein/php_files

<Location /php-source>
 DAV On

    ForceType text/plain
</Location>
```

With this setup, `http://example.com/phparea` can be used to access the output of the PHP scripts, and `http://example.com/php-source` can be used with a DAV client to manipulate them.

Dav 指令

说明	Enable WebDAV HTTP methods
语法	`Dav On|Off|provider-name`
默认值	`Dav Off`
作用域	directory
状态	扩展(E)
模块	mod_dav

Use the `Dav` directive to enable the WebDAV HTTP methods for the given container:

```
     <Location /foo>

 Dav On
     </Location>
```

The value `On` is actually an alias for the default provider `filesystem` which is served by the `mod_dav_fs` module. Note, that once you have DAV enabled for some location, it *cannot* be disabled for sublocations. For a complete configuration example have a look at the section above.

Do not enable WebDAV until you have secured your server. Otherwise everyone will be able to distribute files on your system.

DavDepthInfinity 指令

说明	Allow PROPFIND, Depth: Infinity requests
语法	`DavDepthInfinity on|off`
默认值	`DavDepthInfinity off`
作用域	server config, virtual host, directory
状态	扩展(E)
模块	mod_dav

Use the `DavDepthInfinity` directive to allow the processing of `PROPFIND` requests containing the header 'Depth: Infinity'. Because this type of request could constitute a denial-of-service attack, by default it is not allowed.

DavMinTimeout 指令

说明	Minimum amount of time the server holds a lock on a DAV resource
语法	`DavMinTimeout seconds`
默认值	`DavMinTimeout 0`
作用域	server config, virtual host, directory
状态	扩展(E)
模块	mod_dav

When a client requests a DAV resource lock, it can also specify a time when the lock will be automatically removed by the server. This value is only a request, and the server can ignore it or inform the client of an arbitrary value.

Use the `DavMinTimeout` directive to specify, in seconds, the minimum lock timeout to return to a client. Microsoft Web Folders defaults to a timeout of 120 seconds; the `DavMinTimeout` can override this to a higher value (like 600 seconds) to reduce the chance of the client losing the lock due to network latency.

示例

```
     <Location /MSWord>

DavMinTimeout 600
     </Location>
```

Apache模块 mod_dav_fs

说明	为 `mod_dav` 访问服务器上的文件系统提供支持
状态	扩展(E)
模块名	dav_fs_module
源文件	mod_dav_fs.c

概述

This module *requires* the service of `mod_dav`. It acts as a support module for `mod_dav` and provides access to resources located in the server's file system. The formal name of this provider is `filesystem`. `mod_dav` backend providers will be invoked by using the `Dav` directive:

示例

```
Dav filesystem
```

Since `filesystem` is the default provider for `mod_dav`, you may simply use the value `On` instead.

DavLockDB 指令

说明	Location of the DAV lock database
语法	`DavLockDB file-path`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_dav_fs

Use the `DavLockDB` directive to specify the full path to the lock database, excluding an extension. If the path is not absolute, it will be taken relative to `ServerRoot`. The implementation of `mod_dav_fs` uses a SDBM database to track user locks.

示例

```
DavLockDB var/DavLock
```

The directory containing the lock database file must be writable by the `User` 和 `Group` under which Apache is running. For security reasons, you should create a directory for this purpose rather than changing the permissions on an existing directory. In the above example, Apache will create files in the `var/` directory under the `ServerRoot` with the base filename `DavLock` and extension name chosen by the server.

Apache模块 mod_dav_lock

说明	为 `mod_dav` 锁定服务器上的文件提供支持
状态	扩展(E)
模块名	dav_lock_module
源文件	mod_dav_lock.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

This module implements a generic locking API which can be used by any backend provider of `mod_dav` . It *requires* at least the service of `mod_dav` . But without a backend provider which makes use of it, it's useless and should not be loaded into the server. A sample backend module which actually utilizes `mod_dav_lock` , is mod_dav_svn, the subversion provider module.

Note that `mod_dav_fs` does *not* need this generic locking module, because it uses it's own more specialized version.

In order to make `mod_dav_lock` functional, you just have to specify the location of the lock database using the `DavGenericLockDB` directive described below.

Developer's Note

In order to retrieve the pointer to the locking provider function, you have to use the `ap_lookup_provider` API with the arguments `dav-lock` , `generic` 和 `0` .

DavGenericLockDB 指令

说明	Location of the DAV lock database
语法	`DavGenericLockDB file-path`
作用域	server config, virtual host, directory
状态	扩展(E)
模块	mod_dav_lock

Use the `DavGenericLockDB` directive to specify the full path to the lock database, excluding an extension. If the path is not absolute, it will be taken relative to `ServerRoot`. The implementation of `mod_dav_lock` uses a SDBM database to track user locks.

示例

```
DavGenericLockDB var/DavLock
```

The directory containing the lock database file must be writable by the `User` 和 `Group` under which Apache is running. For security reasons, you should create a directory for this purpose rather than changing the permissions on an existing directory. In the above example, Apache will create files in the `var/` directory under the `ServerRoot` with the base filename `DavLock` and extension name chosen by the server.

Apache模块 mod_dbd

说明	管理SQL数据库连接，为需要数据库功能的模块提供支持
状态	扩展(E)
模块名	dbd_module
源文件	mod_dbd.c
兼容性	Version 2.1 及以后的版本中可用

概述

`mod_dbd` manages SQL database connections using apr_dbd. It provides database connections on request to modules requiring SQL database functions, and takes care of managing databases with optimal efficiency and scalability for both threaded and non-threaded MPMs.

Connection Pooling

This module manages database connections, in a manner optimised for the platform. On non-threaded platforms, it provides a persistent connection in the manner of classic LAMP (Linux, Apache, Mysql, Perl/PHP/Python). On threaded platform, it provides an altogether more scalable and efficient *connection pool*, as described in this article at ApacheTutor. `mod_dbd` supersedes the modules presented in that article.

Apache DBD API

`mod_dbd` exports five functions for other modules to use. The API is as follows:

```
typedef struct {
    apr_dbd_t *handle;
    apr_dbd_driver_t *driver;
    apr_hash_t *prepared;
} ap_dbd_t;

/* Export functions to access the database */

/* acquire a connection that MUST be explicitly closed.
 * Returns NULL on error
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_open(apr_pool_t*, server_rec*);

/* release a connection acquired with ap_dbd_open */
AP_DECLARE(void) ap_dbd_close(server_rec*, ap_dbd_t*);

/* acquire a connection that will have the lifetime of a request
 * and MUST NOT be explicitly closed.  Return NULL on error.
 * This is the preferred function for most applications.
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_acquire(request_rec*);

/* acquire a connection that will have the lifetime of a connection
 * and MUST NOT be explicitly closed.  Return NULL on error.
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_cacquire(request_rec*);

/* Prepare a statement for use by a client module */
AP_DECLARE(void) ap_dbd_prepare(server_rec*, const char*, const char*);

/* Also export them as optional functions for modules that prefer it */
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_open, (apr_pool_t*, server_rec*));
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_close, (server_rec*, ap_dbd_t*));
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_acquire, (request_rec*));
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_cacquire, (conn_rec*));
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_prepare, (server_rec*, const char*, const char*));
```

SQL Prepared Statements

`mod_dbd` supports SQL prepared statements on behalf of modules that may wish to use them. Each prepared statement must be assigned a name (label), and they are stored in a hash: the `prepared` field of an `ap_dbd_t`. Hash entries are of type `apr_dbd_prepared_t` and can be used in any of the apr_dbd prepared statement SQL query or select commands.

It is up to dbd user modules to use the prepared statements and document what statements can be specified in httpd.conf, or to provide their own directives and use `ap_dbd_prepare`.

DBDExptime 指令

说明	Keepalive time for idle connections
语法	`DBDExptime time-in-seconds`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_dbd

Set the time to keep idle connections alive where the number of connections specified in DBDKeep has been exceeded (threaded platforms only).

DBDKeep 指令

说明	Maximum sustainednumber of connections
语法	`DBDKeep number`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_dbd

Set the maximum number of connections per process to be sustained, other than for handling peak demand (threaded platforms only).

DBDMax 指令

说明	Maximum number of connections
语法	`DBDMax number`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_dbd

Set the hard maximum number of connections per process (threaded platforms only).

DBDMin 指令

说明	Minimum number of connections
语法	`DBDMin number`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_dbd

Set the minimum number of connections per process (threaded platforms only).

DBDParams 指令

说明	Parameters for database connection
语法	`DBDParams param1=value1[,param2=value2]`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_dbd

As required by the underlying driver. Typically this will be used to pass whatever cannot be defaulted amongst username, password, database name, hostname and port number for connection.

DBDPersist 指令

说明	Whether to use persistent connections
语法	`DBDPersist 0|1`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_dbd

If set to 0, persistent and pooled connections are disabled. A new database connection is opened when requested by a client, and closed immediately on release. This option is for debugging and low-usage servers.

The default is to enable a pool of persistent connections (or a single LAMP-style persistent connection in the case of a non-threaded server), and should almost always be used in operation.

DBDPrepareSQL 指令

说明	Define an SQL prepared statement
语法	`DBDPrepareSQL "SQL statement" label`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_dbd

For modules such as authentication that use repeatedly use a single SQL statement, optimum performance is achieved by preparing the statement at startup rather than every time it is used. This directive prepares an SQL statement and assigns it a label.

DBDriver 指令

说明	Specify an SQL driver
语法	`DBDriver name`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_dbd

Selects an apr_dbd driver by name. The driver must be installed on your system (on most systems, it will be a shared object or dll). For example, `DBDriver mysql` will select the MySQL driver in apr_dbd_mysql.so.

Apache模块 mod_deflate

说明	压缩发送给客户端的内容
状态	扩展(E)
模块名	deflate_module
源文件	mod_deflate.c

概述

`mod_deflate` 模块提供了 `DEFLATE` 输出过滤器，允许服务器在将输出内容发送到客户端以前进行压缩，以节约带宽。

配置举例

这是一个针对心急者的示范配置：

仅仅压缩少数几种类型

```
AddOutputFilterByType DEFLATE text/html text/plain text/xml
```

以下允许压缩更多内容的配置更加复杂。除非你明白所有的配置细节，否则请不要使用。

Compress everything except images

```
    <Location />
# 插入过滤器
        SetOutputFilter DEFLATE
        # Netscape 4.x 有一些问题...
        BrowserMatch ^Mozilla/4         gzip-only-text/html
        # Netscape 4.06-4.08 有更多的问题
        BrowserMatch ^Mozilla/4\.0[678] no-gzip
        # MSIE 会伪装成 Netscape ，但是事实上它没有问题
        BrowserMatch \bMSIE             !no-gzip !gzip-only-text/html
        # 不压缩图片
        SetEnvIfNoCase Request_URI \
\.(?:gif|jpe?g|png)$ no-gzip dont-vary
        # 确保代理不会发送错误的内容
        Header append Vary User-Agent env=!dont-vary
    </Location>
```

启用压缩

输出压缩

压缩是由 `DEFLATE` 过滤器实现的。下面的指令会对其所在容器中的文档启用压缩：

```
        SetOutputFilter DEFLATE
```

一些流行的浏览器不能正确处理所有压缩内容，因此你可能需要将 `gzip-only-text/html` 标记设为" `1` "来仅仅允许压缩html文件(见下面)。如果你设置了" `1` "以外的任何值，都将被忽略。

如果你想将压缩限制在几种特定的MIME类型上，可以使用 `AddOutputFilterByType` 指令。下面的例子仅仅允许对html文档进行压缩：

```
        <Directory "/your-server-root/manual">
AddOutputFilterByType DEFLATE text/html
        </Directory>
```

对于那些不能正确处理所有压缩内容的浏览器，可以使用 `BrowserMatch` 指令针对特定的浏览器设置 `no-gzip` 标记以取消压缩。为了取得更好的效果，你可以将 `no-gzip` 和 `gzip-only-text/html` 配合使用。在这种情况下，下面的设置将会覆盖上面的设置。看看从配置示例中摘录的片断：

```
        BrowserMatch ^Mozilla/4         gzip-only-text/html
        BrowserMatch ^Mozilla/4\.0[678] no-gzip
        BrowserMatch \bMSIE             !no-gzip !gzip-only-text/html
```

第一条指令表示如果 `User-Agent` 字符串表示它是一个Navigator 4.x的浏览器，这种浏览器不能正确处理除 `text/html` 之外的所有类型。而4.06, 4.07, 4.08版的Navigator完全不能处理任何压缩内容，因此第二条指令对这些浏览器完全禁用压缩。

第三个 `BrowserMatch` 指令修正了上面两条对浏览器的推测，因为微软的IE也将它自己标识成"Mozilla/4"但是它实际上能够处理所有的压缩内容。因此又在 `User-Agent` 头中额外匹配了字符串"MSIE"(" `\b` "表示"单词边界")，并且取消了前面的限制。

注意

`DEFLATE` 过滤器总是在类似于PHP或SSI之类的资源过滤器之后插入，它永远不会触及到内部请求。

注意

通过 `SetEnv` 设置 `force-gzip` 环境变量将会忽略浏览器的"accept-encoding"，始终发送经过压缩的内容。

输出解压

`mod_deflate` 模块还提供了一个解压gzip格式的应答体的功能。为了激活这个特性你必须使用 `SetOutputFilter` 或 `AddOutputFilter` 指令将 `INFLATE` 过滤器插入到输入过滤器链：

```
        <Location /dav-area>
  ProxyPass http://example.com/
          SetOutputFilter INFLATE
        </Location>
```

这个例子将会解压来自example.com的输出，这样其它过滤器就可以做进一步的处理了。

输入解压

`mod_deflate` 模块还提供了一个解压gzip格式的请求体的功能。为了激活这个特性你必须使用 `SetInputFilter` 或 `AddInputFilter` 指令将 `DEFLATE` 过滤器插入到输入过滤器链。例如：

```
        <Location /dav-area>
SetInputFilter DEFLATE
        </Location>
```

这样，如果包含"`Content-Encoding: gzip`"头的请求体将会被自动解压。极少有浏览器压缩请求体。然而有些程序的确这么做了，比如一些WebDAV客户端程序。

注意 Content-Length

如果你自己处理请求体，请注意`Content-Length`头仅仅表示客户端输入的数据长度，而不是解压后的实际数据长度。

代理服务器

`mod_deflate`模块发送一个"`Vary: Accept-Encoding`"HTTP应答头以提醒代理服务器：只对发送了正确"`Accept-Encoding`"头的客户端发送缓存的应答。这样可以防止不能正确处理压缩内容的浏览器接受到经过压缩的内容。

如果你按照某些特殊的条件拒绝了某些客户端的访问(比如`User-Agent`头)，你必须手动配置一个额外的`Vary`头提醒代理服务器做额外的限制。比如，在一个典型的配置中的某处，如果额外的`DEFLATE`过滤器是否生效取决于`User-Agent`头，你应当在此处添加：

```
        Header append Vary User-Agent
```

如果依照除请求头以外的其他条件决定是否使用压缩(例如：HTTP版本)，你必须设置`Vary`头的值为"`*`"来完全阻止代理服务器的缓存。

示例

```
        Header set Vary *
```

DeflateBufferSize 指令

说明	用于zlib一次压缩的片断大小(字节)
语法	`DeflateBufferSize value`
默认值	`DeflateBufferSize 8096`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_deflate

`DeflateBufferSize` 指令定义了zlib一次压缩的片断的字节数。

DeflateCompressionLevel 指令

说明	将输出内容压缩的程度
语法	`DeflateCompressionLevel value`
默认值	`Zlib的默认值`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_deflate
兼容性	仅在 Apache 2.0.45 及以后的版本中可用

`DeflateCompressionLevel` 指令设置压缩程度，越高的压缩程度就会有越好的压缩效果，同时也意味着占用越多的CPU资源。

取值范围在 1(最低压缩率) 到 9(最高压缩率)之间。

DeflateFilterNote 指令

说明	在日志中放置压缩率标记
语法	`DeflateFilterNote [type] notename`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_deflate
兼容性	type仅在2.0.45以后版本中可用

`DeflateFilterNote` 指令指定将一个指示压缩率的标记附加在请求之后。notename就表示这个标记的名字。你可以为了某种统计目的将这个标记的名字添加到访问日志中。

示例

```
DeflateFilterNote ratio
LogFormat '"%r" %b (%{ratio}n) "%{User-agent}i"' deflate
CustomLog logs/deflate_log deflate
```

如果你想从日志中得到更多精确的数据，可以使用type参数指定notename标记所记录的数据类型。type的取值范围如下：

`Input`

在标记中存储过滤器输入流的字节数。

`Output`

在标记中存储过滤器输出流的字节数。

`Ratio`

在标记中存储过滤器的压缩比(`输出/输入*100`)。这是type的默认值。

于是，就可以这样记录：

Accurate Logging

```
DeflateFilterNote Input instream
DeflateFilterNote Output outstream
DeflateFilterNote Ratio ratio
LogFormat '"%r" %{outstream}n/%{instream}n (%{ratio}n%%)' deflate
CustomLog logs/deflate_log deflate
```

参见

- `mod_log_config`

DeflateMemLevel 指令

说明	zlib在压缩时最多可以使用多少内存
语法	`DeflateMemLevel value`
默认值	`DeflateMemLevel 9`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_deflate

`DeflateMemLevel` 指令指定zlib在压缩时最多可以使用多少内存(取值范围在1到9之间)。

DeflateWindowSize 指令

说明	Zlib压缩窗口(compression window)的大小
语法	`DeflateWindowSize value`
默认值	`DeflateWindowSize 15`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_deflate

`DeflateWindowSize` 指令设定zlib压缩窗口(compression window)的大小(取值范围在1到15之间)。通常窗口越大压缩效果越好。

Apache模块 mod_dir

说明	指定目录索引文件以及为目录提供"尾斜杠"重定向
状态	基本(B)
模块名	dir_module
源文件	mod_dir.c

概述

目录的索引可以有两个来源：

- 一个由用户编写的文件，通常叫：`index.html` 。`mod_dir` 提供的 `DirectoryIndex` 指令用于设置这个文件名。
- 由服务器产生的一个列表。该功能由 `mod_autoindex` 提供。

这两个功能是相互独立的，所以你可以完全去除或替换索引的自动生成。

因为对目录的请求需要以一个"/"结尾，所以当服务器接收到对 `http://servername/foo/dirname` 的请求时，若 `dirname` 是一个目录，则 `mod_dir` 将会将其重定向到 `http://servername/foo/dirname/` 。

DirectoryIndex 指令

说明	当客户端请求一个目录时寻找的资源列表
语法	`DirectoryIndex local-url [local-url] ...`
默认值	`DirectoryIndex index.html`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_dir

`DirectoryIndex` 指令设置了当客户端在请求的目录名的末尾刻意添加一个"/"以表示请求该目录的索引时，服务器需要寻找的资源列表。Local-url(%已解码的)是一个相对于被请求目录的文档的URL(通常是那个目录中的一个文件)。可以指定多个URL，服务器将返回最先找到的那一个。若一个也没有找到，并且那个目录设置了 `Indexes` 选项，服务器将会自动产生一个那个目录中的资源列表。

示例

```
DirectoryIndex index.html
```

上例配置指示对 `http://myserver/docs/` 的请求返回 `http://myserver/docs/index.html` (若存在)，或返回该目录下所有资源的列表。

注意，指定的文档不一定必须位于被请求的目录下，也可以指定一个绝对URL来指向其他位置：

```
DirectoryIndex index.html index.txt  /cgi-bin/index.pl
```

这样的设置将导致在 `index.html` 或 `index.txt` 都不存在的情况下执行CGI脚本 `/cgi-bin/index.pl` 。

DirectorySlash 指令

说明	打开或关闭目录结尾斜线(/)自动补全功能
语法	`DirectorySlash On|Off`
默认值	`DirectorySlash On`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_dir
兼容性	仅在 Apache 2.0.51 及以后的版本中可用

`DirectorySlash` 指令决定 `mod_dir` 是否通过在请求的URL结尾补上"/"使其重定向到其所指向的目录。

典型的，如果用户请求一个结尾没有"/"的资源，并且该资源指向一个目录， `mod_dir` 将通过在请求的URL结尾补上"/"使其重定向到其所指向的目录。默认开启自动补全功能有以下原因：

- 用户最后使用了规范的URL来请求资源。
- `mod_autoindex` 将会正确工作。因为它不会自动补全路径，所以将会指向错误的路径。
- `DirectoryIndex` 将只评估有"/"结尾的目录。
- html页面中的相对URL引用将会正确工作。

如果你不希望这个自动补全功能生效，并且不在乎上述原因，你可以关闭它：

```
        # 请参见下面的安全警告

        <Location /some/path>

DirectorySlash Off

            SetHandler some-handler
        </Location>
```

安全警告

关闭目录自动重定向可能会导致信息泄漏。考虑 `mod_autoindex` 被激活(`Options +Indexes`)并且 `DirectoryIndex` 也正确设置到一个资源(比如： `index.html`)同时没有其他处理器用于URL的情况。此时，以"/"结尾的URL将得到 `index.html` 文件，而不以**"/"**结尾的请求将得到目录列表。

Apache模块 mod_disk_cache

说明	基于磁盘的缓冲管理器
状态	扩展(E)
模块名	disk_cache_module
源文件	mod_disk_cache.c

概述

`mod_disk_cache` implements a disk based storage manager. It is primarily of use in conjunction `mod_cache` .

Content is stored in and retrieved from the cache using URI based keys. Content with access protection is not cached.

`htcacheclean` can be used to maintain the cache size at a maximum level.

注意：

`mod_disk_cache` requires the services of `mod_cache` .

CacheDirLength 指令

说明	The number of characters in subdirectory names
语法	`CacheDirLength length`
默认值	`CacheDirLength 2`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_disk_cache

`CacheDirLength` directive sets the number of characters for each subdirectory name in the cache hierarchy.

The result of `CacheDirLevels` * `CacheDirLength` must not be higher than 20.

```
CacheDirLength  4
```

CacheDirLevels 指令

说明	The number of levels of subdirectories in the cache.
语法	`CacheDirLevels levels`
默认值	`CacheDirLevels 3`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_disk_cache

`CacheDirLevels` directive sets the number of subdirectory levels in the cache. Cached data will be saved this many directory levels below the `CacheRoot` directory.

The result of `CacheDirLevels` * `CacheDirLength` must not be higher than 20.

```
CacheDirLevels  5
```

CacheMaxFileSize 指令

说明	The maximum size (in bytes) of a document to be placed in the cache
语法	`CacheMaxFileSize bytes`
默认值	`CacheMaxFileSize 1000000`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_disk_cache

`CacheMaxFileSize` directive sets the maximum size, in bytes, for a document to be considered for storage in the cache.

```
CacheMaxFileSize 64000
```

CacheMinFileSize 指令

说明	The minimum size (in bytes) of a document to be placed in the cache
语法	`CacheMinFileSize bytes`
默认值	`CacheMinFileSize 1`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_disk_cache

`CacheMinFileSize` directive sets the minimum size, in bytes, for a document to be considered for storage in the cache.

```
CacheMinFileSize 64
```

CacheRoot 指令

说明	The directory root under which cache files are stored
语法	`CacheRoot directory`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_disk_cache

`CacheRoot` directive defines the name of the directory on the disk to contain cache files. If the `mod_disk_cache` module has been loaded or compiled in to the Apache server, this directive *must* be defined. Failing to provide a value for `CacheRoot` will result in a configuration file processing error. The `CacheDirLevels` 和 `CacheDirLength` directives define the structure of the directories under the specified root directory.

```
CacheRoot c:/cacheroot
```

Apache模块 mod_dumpio

说明	将所有I/O操作转储到错误日志中
状态	扩展(E)
模块名	dumpio_module
源文件	mod_dumpio.c

概述

`mod_dumpio` 允许你记录所有Apache接收到的输入和发送的输出到错误日志(通常是error.log)中。

记录数据的时刻被设计为恰好发生在SSL解码(输入)之后和SSL编码之前(输出)。正如你所预料到的，这么做会导致在日志中写入及其海量的数据，只建议你在发现问题并进行调试的时候使用。

启用dumpio支持

要启用这个模块，必须先编译完成它，然后在配置文件中加载，然后用下面列出的指令启用或者禁用记录功能。

DumpIOInput 指令

说明	将所有输入的内容记录到错误日志中
语法	`DumpIOInput On|Off`
默认值	`DumpIOInput Off`
作用域	server config
状态	扩展(E)
模块	mod_dumpio
兼容性	仅在 Apache 2.1.3 及以后的版本中可用

将所有输入的内容记录到错误日志中

示例

```
DumpIOInput On
```

DumpIOOutput 指令

说明	将所有输出的内容记录到错误日志中
语法	`DumpIOOutput On|Off`
默认值	`DumpIOOutput Off`
作用域	server config
状态	扩展(E)
模块	mod_dumpio
兼容性	仅在 Apache 2.1.3 及以后的版本中可用

将所有输出的内容记录到错误日志中

示例

```
DumpIOOutput On
```

Apache模块 mod_echo

说明	一个很简单的协议演示模块
状态	实验(X)
模块名	echo_module
源文件	mod_echo.c
兼容性	仅在 Apache 2.0 及以后的版本中可用

概述

This module provides an example protocol module to illustrate the concept. It provides a simple echo server. Telnet to it and type stuff, and it will echo it.

ProtocolEcho 指令

说明	Turn the echo server on or off
语法	`ProtocolEcho On|Off`
作用域	server config, virtual host
状态	实验(X)
模块	mod_echo
兼容性	ProtocolEcho is only available in 2.0 及以后的版本中可用

`ProtocolEcho` directive enables or disables the echo server.

示例

```
ProtocolEcho On
```

Apache模块 mod_env

说明	允许Apache修改或清除传送到CGI脚本和SSI页面的环境变量
状态	基本(B)
模块名	env_module
源文件	mod_env.c

概述

本模块用于控制传送给CGI脚本和SSI页面的环境变量。所传送的环境变量可以来自调用 `httpd` 进程的shell，或者来自在配置过程中所设定(set)或撤销(unset)的变量。

PassEnv 指令

说明	传送shell中的环境变量
语法	`PassEnv env-variable [env-variable] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_env

从调用 `httpd` 进程所在的shell中，指定一个或者更多个环境变量，传送到CGI脚本和SSI页面。例如：

示例

```
PassEnv LD_LIBRARY_PATH
```

SetEnv 指令

说明	设置环境变量
语法	`SetEnv env-variable value`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_env

设置一个环境变量，该变量将会传送到CGI脚本和SSI页面。例如：

示例

```
SetEnv SPECIAL_PATH /foo/bin
```

UnsetEnv 指令

说明	删除一个环境变量
语法	`UnsetEnv env-variable [env-variable] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_env

在传送到CGI脚本和SSI页面的环境中，删除一个或者多个环境变量。例如：

示例

```
UnsetEnv LD_LIBRARY_PATH
```

Apache模块 mod_example

说明	一个很简单的Apache模块API演示模块
状态	实验(X)
模块名	example_module
源文件	mod_example.c

概述

Some files in the `modules/experimental` directory under the Apache distribution directory tree are provided as an example to those that wish to write modules that use the Apache API.

The main file is `mod_example.c`, which illustrates all the different callback mechanisms and call syntaxes. By no means does an add-on module need to include routines for all of the callbacks - quite the contrary!

The example module is an actual working module. If you link it into your server, enable the "example-handler" handler for a location, and then browse to that location, you will see a display of some of the tracing the example module did as the various callbacks were made.

Compiling the example module

To include the example module in your server, follow the steps below:

1. Run `configure` with `--enable-example` option.
2. Make the server (run " `make` ").

To add another module of your own:

1. `cp modules/experimental/mod_example.c modules/new_module/_mod_myexample.c_`
2. Modify the file.
3. Create `modules/new_module/config.m4`.
    i. Add `APACHE_MODPATH_INIT(new_module)`.
    ii. Copy APACHE_MODULE line with "example" from `modules/experimental/config.m4`.
    iii. Replace the first argument "example" with *myexample*.
    iv. Replace the second argument with brief description of your module. It will be used in `configure --help`.

   v. If your module needs additional C compiler flags, linker flags or libraries, add them to CFLAGS, LDFLAGS and LIBS accordingly. See other `config.m4` files in modules directory for examples.
   vi. Add `APACHE_MODPATH_FINISH`.
4. Create `module/new_module/Makefile.in`. If your module doesn't need special build instructions, all you need to have in that file is `include $(top_srcdir)/build/special.mk`.
5. Run ./buildconf from the top-level directory.
6. Build the server with --enable-myexample

Using the `mod_example` Module

To activate the example module, include a block similar to the following in your `httpd.conf` file:

```
<Location /example-info>
    SetHandler example-handler
</Location>
```

As an alternative, you can put the following into a `.htaccess` file and then request the file "test.example" from that location:

```
AddHandler example-handler .example
```

After reloading/restarting your server, you should be able to browse to this location and see the brief display mentioned earlier.

Example 指令

说明	Demonstration directive to illustrate the Apache module API
语法	`Example`
作用域	server config, virtual host, directory, .htaccess
状态	实验(X)
模块	mod_example

`Example` directive just sets a demonstration flag which the example module's content handler displays. It takes no arguments. If you browse to an URL to which the example content-handler applies, you will get a display of the routines within the module and how and

in what order they were called to service the document request. The effect of this directive one can observe under the point " `Example directive declared here: YES/NO` ".

Apache模块 mod_expires

说明	允许通过配置文件控制HTTP的" `Expires` "和" `Cache-Control` "头内容
状态	扩展(E)
模块名	expires_module
源文件	mod_expires.c

概述

这个模块控制服务器应答时的 `Expires` 头内容和 `Cache-Control` 头的 `max-age` 指令。有效期(expiration date)可以设置为相对于源文件的最后修改时刻或者客户端的访问时刻。

这些HTTP头向客户端表明了文档的有效性和持久性。如果有缓存，文档就可以从缓存(除已经过期)而不是从服务器读取。接着，客户端考察缓存中的副本，看看是否过期或者失效，以决定是否必须从服务器获得更新。

要修改 `Cache-Control` 头中 `max-age` (参见RFC 2616 section 14.9)项之外的内容，你还可以使用 `Header` 指令。

Alternate(交替/轮流) Interval(间隔) Syntax(语法)

`ExpiresDefault` 和 `ExpiresByType` 指令同样能够用易懂的语法格式进行定义：

```
ExpiresDefault "<base> [plus] {<num>
<type>}*"

ExpiresByType type/encoding "<base> [plus]
{<num> <type>}*"
```

其中<base>是下列之一：

- `access`
- `now` (等价于' `access` ')
- `modification`

`plus` 关键字是可选的。<num>必须是整数[可以被 `atoi()` 接受的]，<type>是下列之一：

- `years`
- `months`
- `weeks`
- `days`

- `hours`
- `minutes`
- `seconds`

例如，下列3个指令都表示文档默认的有效期是一个月：

```
ExpiresDefault "access plus 1 month"

ExpiresDefault "access plus 4 weeks"

ExpiresDefault "access plus 30 days"
```

有效期可以通过增加"<num> <type>"子句进一步调整：

```
ExpiresByType text/html "access plus 1 month 15
days 2 hours"

ExpiresByType image/gif "modification plus 5 hours 3
minutes"
```

注意，如果你使用基于最后修改日期的设置，"Expires:"头将不会被添加到那些并非来自于磁盘文件的内容。这是因为这些内容并不存在"最后修改时间"的属性。

ExpiresActive 指令

说明	启用或禁用产生" `Expires:` "和" `Cache-Control:` "头的功能
语法	`ExpiresActive On|Off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	扩展(E)
模块	mod_expires

这个指令对其作用范围内的文档启用或禁用产生 `Expires` 和 `Cache-Control` 头的功能。若设置为 `Off` 则不会为其作用范围内的任何文档生成 `Expires` 和 `Cache-Control` 头(除非被更低一层的规则改写，比如 `.htaccess` 文件)。若设置为 `On` 则会按照 `ExpiresByType` 和 `ExpiresDefault` 指令定义的标准为其作用范围内的文档生成 `Expires` 和 `Cache-Control` 头。

注意，这个指令并不保证 `Expires` 或 `Cache-Control` 头一定会产生。如果定义的标准不规范，将不会产生这两个头，其效果是好像从未设置过这个指令一样。

ExpiresByType 指令

说明	由MIME类型配置的 `Expires` 头的值
语法	`ExpiresByType MIME-type <code>seconds`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	扩展(E)
模块	mod_expires

这个指令定义了为指定MIME类型的文档(如 `text/html`)生成的 `Expires` 头的值和 `Cache-Control` 头的 `max-age` 指令。seconds参数设置了添加到基准时间以构造有效期限的秒数。 `Cache-Control: max-age` 的计算方法是从有效期减去当前请求时间并转化为秒数。

基准时刻可以是源文件的最后修改时刻或者客户端对源文件的访问时刻，至于使用那一个则由 `<code>` 指定。" `M` "表示源文件的最后修改时刻，" `A` "表示客户端对源文件的访问时刻。需要注意的是 `<code>` 和 `seconds` 之间没有空格。

这两种基准的差别是很微妙的。如果使用" `M` "，所有当前缓存中的文档副本都将在同一时刻过期，这个可能对定期更新的URL(比如位于同一位置的每周通告)很有好处。如果使用" `A` "，则每个客户端所得到的有效期是不一样的，这个可能对那些几乎不更新的图片文件很有好处，特别是对于一组都引用了相同图片的相关文档。

示例：

```
# 启用有效期控制
ExpiresActive On
# GIF有效期为1个月
ExpiresByType image/gif A2592000
# HTML文档的有效期是最后修改时刻后的一星期
ExpiresByType text/html M604800
```

注意，这个指令只有在" `ExpiresActive On` "的条件下才有效。它只对指定的MIME类型文档改写由 `ExpiresDefault` 指令设置的有效期。

你也可以使用前面讲述的alternate syntax指定有效期的计算方法。

ExpiresDefault 指令

说明	默认有效期的计算方法
语法	`ExpiresDefault <code>seconds`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	扩展(E)
模块	mod_expires

该指令设置了其作用范围内的所有文档的默认有效期的计算方法，它可以被 `ExpiresByType` 指令基于MIME类型被改写。详情请参见 `ExpiresByType` 指令和那个alternate syntax的描述。

Apache模块 mod_ext_filter

说明	使用外部程序作为过滤器
状态	扩展(E)
模块名	ext_filter_module
源文件	mod_ext_filter.c

概述

`mod_ext_filter` presents a simple and familiar programming model for 过滤器. With this module, a program which reads from stdin and writes to stdout (i.e., a Unix-style filter command) can be a filter for Apache. This filtering mechanism is much slower than using a filter which is specially written for the Apache API and runs inside of the Apache server process, but it does have the following benefits:

- the programming model is much simpler
- any programming/scripting language can be used, provided that it allows the program to read from standard input and write to standard output
- existing programs can be used unmodified as Apache filters

Even when the performance characteristics are not suitable for production use, `mod_ext_filter` can be used as a prototype environment for filters.

Examples

Generating HTML from some other type of response

```
        # mod_ext_filter directive to define a filter
        # to HTML-ize text/c files using the external
        # program /usr/bin/enscript, with the type of
        # the result set to text/html
        ExtFilterDefine c-to-html mode=output \
intype=text/c outtype=text/html \
          cmd="/usr/bin/enscript --color -W html -Ec -o - -"
        <Directory "/export/home/trawick/apacheinst/htdocs/c">
# core directive to cause the new filter to
          # be run on output
          SetOutputFilter c-to-html
          # mod_mime directive to set the type of .c
          # files to text/c
          AddType text/c .c
          # mod_ext_filter directive to set the debug
          # level just high enough to see a log message
          # per request showing the configuration in force
          ExtFilterOptions DebugLevel=1
        </Directory>
```

Implementing a content encoding filter

Note: this gzip example is just for the purposes of illustration. Please refer to `mod_deflate` for a practical implementation.

```
        # mod_ext_filter directive to define the external filter
        ExtFilterDefine gzip mode=output cmd=/bin/gzip
        <Location /gzipped>
# core directive to cause the gzip filter to be
          # run on output
          SetOutputFilter gzip
          # mod_header directive to add
          # "Content-Encoding: gzip" header field
          Header set Content-Encoding gzip
        </Location>
```

Slowing down the server

```
        # mod_ext_filter directive to define a filter

        # which runs everything through cat; cat doesn't

        # modify anything; it just introduces extra pathlength

        # and consumes more resources

        ExtFilterDefine slowdown mode=output cmd=/bin/cat \

preservescontentlength

        <Location />

# core directive to cause the slowdown filter to

          # be run several times on output

          #

          SetOutputFilter slowdown;slowdown;slowdown
        </Location>
```

Using sed to replace text in the response

```
        # mod_ext_filter directive to define a filter which

        # replaces text in the response

        #

        ExtFilterDefine fixtext mode=output intype=text/html \

cmd="/bin/sed s/verdana/arial/g"

        <Location />

# core directive to cause the fixtext filter to

          # be run on output

          SetOutputFilter fixtext
        </Location>
```

Tracing another filter

```
        # Trace the data read and written by mod_deflate

        # for a particular client (IP 192.168.1.31)

        # experiencing compression problems.

        # This filter will trace what goes into mod_deflate.

        ExtFilterDefine tracebefore \

cmd="/bin/tracefilter.pl /tmp/tracebefore" \

          EnableEnv=trace_this_client

        # This filter will trace what goes after mod_deflate.

        # Note that without the ftype parameter, the default

        # filter type of AP_FTYPE_RESOURCE would cause the

        # filter to be placed *before* mod_deflate in the filter

        # chain.  Giving it a numeric value slightly higher than

        # AP_FTYPE_CONTENT_SET will ensure that it is placed

        # after mod_deflate.

        ExtFilterDefine traceafter \

cmd="/bin/tracefilter.pl /tmp/traceafter" \

          EnableEnv=trace_this_client ftype=21

        <Directory /usr/local/docs>

SetEnvIf Remote_Addr 192.168.1.31 trace_this_client

          SetOutputFilter tracebefore;deflate;traceafter
        </Directory>
```

Here is the filter which traces the data:

```
        #!/usr/local/bin/perl -w

        use strict;

        open(SAVE, ">$ARGV[0]")

or die "can't open $ARGV[0]: $?";

        while (<STDIN>) {

print SAVE $_;

          print $_;
        }

        close(SAVE);
```

ExtFilterDefine 指令

说明	Define an external filter
语法	`ExtFilterDefine filtername parameters`
作用域	server config
状态	扩展(E)
模块	mod_ext_filter

`ExtFilterDefine` directive defines the characteristics of an external filter, including the program to run and its arguments.

filtername specifies the name of the filter being defined. This name can then be used in `SetOutputFilter` directives. It must be unique among all registered filters. *At the present time, no error is reported by the register-filter API, so a problem with duplicate names isn't reported to the user.*

Subsequent parameters can appear in any order and define the external command to run and certain other characteristics. The only required parameter is `cmd=` . These parameters are:

`cmd=cmdline`

The `cmd=` keyword allows you to specify the external command to run. If there are arguments after the program name, the command line should be surrounded in quotation marks (例如， `cmd="/bin/mypgm arg1 arg2"` .) Normal shell quoting is not necessary since the program is run directly, bypassing the shell. Program arguments are blank-delimited. A backslash can be used to escape blanks which should be part of a program argument. Any backslashes which are part of the argument must be escaped with backslash themselves. In addition to the standard CGI environment variables, DOCUMENT_URI, DOCUMENT_PATH_INFO, and QUERY_STRING_UNESCAPED will also be set for the program.

`mode=mode`

Use `mode=output` (the default) for filters which process the response. Use `mode=input` for filters which process the request. `mode=input` is available in Apache 2.1 and later.

`intype=imt`

This parameter specifies the internet media type (*i.e.*, MIME type) of documents which should be filtered. By default, all documents are filtered. If `intype=` is specified, the filter will be disabled for documents of other types.

`outtype=imt`

This parameter specifies the internet media type (*i.e.*, MIME type) of filtered documents. It is useful when the filter changes the internet media type as part of the filtering operation. By default, the internet media type is unchanged.

`PreservesContentLength`

The `PreservesContentLength` keyword specifies that the filter preserves the content length. This is not the default, as most filters change the content length. In the event that the filter doesn't modify the length, this keyword should be specified.

`ftype=filtertype`

This parameter specifies the numeric value for filter type that the filter should be registered as. The default value, AP_FTYPE_RESOURCE, is sufficient in most cases. If the filter needs to operate at a different point in the filter chain than resource filters, then this parameter will be necessary. See the AP_FTYPE_foo definitions in util_filter.h for appropriate values.

`disableenv=env`

This parameter specifies the name of an environment variable which, if set, will disable the filter.

`enableenv=env`

This parameter specifies the name of an environment variable which must be set, or the filter will be disabled.

ExtFilterOptions 指令

说明	**Configure `mod_ext_filter` options**
语法	`ExtFilterOptions option [option] ...`
默认值	`ExtFilterOptions DebugLevel=0 NoLogStderr`
作用域	directory
状态	扩展(E)
模块	mod_ext_filter

`ExtFilterOptions` directive specifies special processing options for `mod_ext_filter`. Option can be one of

`DebugLevel=n`

The `DebugLevel` keyword allows you to specify the level of debug messages generated by `mod_ext_filter`. By default, no debug messages are generated. This is equivalent to `DebugLevel=0`. With higher numbers, more debug messages are generated, and server

performance will be degraded. The actual meanings of the numeric values are described with the definitions of the DBGLVL_ constants near the beginning of `mod_ext_filter.c` .

Note: The core directive `LogLevel` should be used to cause debug messages to be stored in the Apache error log.

`LogStderr | NoLogStderr`

The `LogStderr` keyword specifies that messages written to standard error by the external filter program will be saved in the Apache error log. `NoLogStderr` disables this feature.

示例

```
ExtFilterOptions LogStderr DebugLevel=0
```

Messages written to the filter's standard error will be stored in the Apache error log. No debug messages will be generated by `mod_ext_filter` .

Apache模块 mod_file_cache

说明	提供文件描述符缓存支持，从而提高**Apache**性能
状态	实验(X)
模块名	file_cache_module
源文件	mod_file_cache.c

概述

This module should be used with care. You can easily create a broken site using `mod_file_cache`, so read this document carefully.

Caching frequently requested files that change very infrequently is a technique for reducing server load. `mod_file_cache` provides two techniques for caching frequently requested *static* files. Through configuration directives, you can direct `mod_file_cache` to either open then `mmap()` a file, or to pre-open a file and save the file's open *file handle*. Both techniques reduce server load when processing requests for these files by doing part of the work (specifically, the file I/O) for serving the file when the server is started rather than during each request.

注意：You cannot use this for speeding up CGI programs or other files which are served by special content handlers. It can only be used for regular files which are usually served by the Apache core content handler.

This module is an extension of and borrows heavily from the `mod_mmap_static` module in Apache 1.3.

Using mod_file_cache

`mod_file_cache` caches a list of statically configured files via `MMapFile` 或 `CacheFile` directives in the main server configuration.

Not all platforms support both directives. For example, Apache on Windows does not currently support the `MMapStatic` directive, while other platforms, like AIX, support both. You will receive an error message in the server error log if you attempt to use an unsupported directive. If given an unsupported directive, the server will start but the file will not be cached. On platforms that support both directives, you should experiment with both to see which works best for you.

MMapFile Directive

`MMapFile` directive of `mod_file_cache` maps a list of statically configured files into memory through the system call `mmap()`. This system call is available on most modern Unix derivates, but not on all. There are sometimes system-specific limits on the size and number of files that can be `mmap()` ed, experimentation is probably the easiest way to find out.

This `mmap()` ing is done once at server start or restart, only. So whenever one of the mapped files changes on the filesystem you *have* to restart the server (see the Stopping and Restarting documentation). To reiterate that point: if the files are modified *in place* without restarting the server you may end up serving requests that are completely bogus. You should update files by unlinking the old copy and putting a new copy in place. Most tools such as `rdist` 和 `mv` do this. The reason why this modules doesn't take care of changes to the files is that this check would need an extra `stat()` every time which is a waste and against the intent of I/O reduction.

CacheFile Directive

`CacheFile` directive of `mod_file_cache` opens an active *handle或file descriptor* to the file (or files) listed in the configuration directive and places these open file handles in the cache. When the file is requested, the server retrieves the handle from the cache and passes it to the `sendfile()` (or `TransmitFile()` on Windows), socket API.

This file handle caching is done once at server start or restart, only. So whenever one of the cached files changes on the filesystem you *have* to restart the server (see the Stopping and Restarting documentation). To reiterate that point: if the files are modified *in place* without restarting the server you may end up serving requests that are completely bogus. You should update files by unlinking the old copy and putting a new copy in place. Most tools such as `rdist` 和 `mv` do this.

注意

Don't bother asking for a directive which recursively caches all the files in a directory. Try this instead... See the `Include` directive, and consider this command:

```
find /www/htdocs -type f -print \

| sed -e 's/.*/mmapfile &/' > /www/conf/mmap.conf
```

CacheFile 指令

说明	Cache a list of file handles at startup time
语法	`CacheFile file-path [file-path] ...`
作用域	server config
状态	实验(X)
模块	mod_file_cache

`CacheFile` directive opens handles to one or more files (given as whitespace separated arguments) and places these handles into the cache at server startup time. Handles to cached files are automatically closed on a server shutdown. When the files have changed on the filesystem, the server should be restarted to to re-cache them.

Be careful with the file-path arguments: They have to literally match the filesystem path Apache's URL-to-filename translation handlers create. We cannot compare inodes or other stuff to match paths through symbolic links *etc.* because that again would cost extra `stat()` system calls which is not acceptable. This module may or may not work with filenames rewritten by `mod_alias` 或 `mod_rewrite` .

示例

```
CacheFile /usr/local/apache/htdocs/index.html
```

MMapFile 指令

说明	Map a list of files into memory at startup time
语法	`MMapFile file-path [file-path] ...`
作用域	server config
状态	实验(X)
模块	mod_file_cache

`MMapFile` directive maps one or more files (given as whitespace separated arguments) into memory at server startup time. They are automatically unmapped on a server shutdown. When the files have changed on the filesystem at least a `HUP` 或 `USR1` signal should be send to the server to re- `mmap()` them.

Be careful with the file-path arguments: They have to literally match the filesystem path Apache's URL-to-filename translation handlers create. We cannot compare inodes or other stuff to match paths through symbolic links *etc.* because that again would cost extra `stat()`

system calls which is not acceptable. This module may or may not work with filenames rewritten by `mod_alias` 或 `mod_rewrite` .

示例

```
MMapFile /usr/local/apache/htdocs/index.html
```

Apache模块 mod_filter

说明	根据上下文实际情况对输出过滤器进行动态配置
状态	基本(B)
模块名	filter_module
源文件	mod_filter.c
兼容性	Version 2.1 及以后的版本中可用

概述

This module enables smart, context-sensitive configuration of output content filters. For example, apache can be configured to process different content-types through different filters, even when the content-type is not known in advance (e.g. in a proxy).

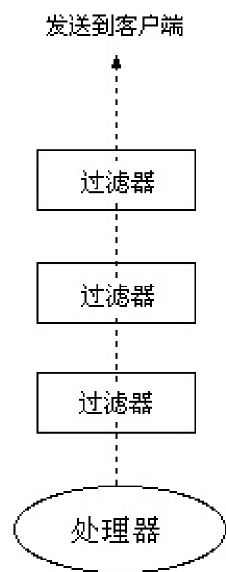
`mod_filter` works by introducing indirection into the filter chain. Instead of inserting filters in the chain, we insert a filter harness which in turn dispatches conditionally to a filter provider. Any content filter may be used as a provider to `mod_filter` ; no change to existing filter modules is required (although it may be possible to simplify them).

Smart Filtering

In the traditional filtering model, filters are inserted unconditionally using `AddOutputFilter` and family. Each filter then needs to determine whether to run, and there is little flexibility available for server admins to allow the chain to be configured dynamically.

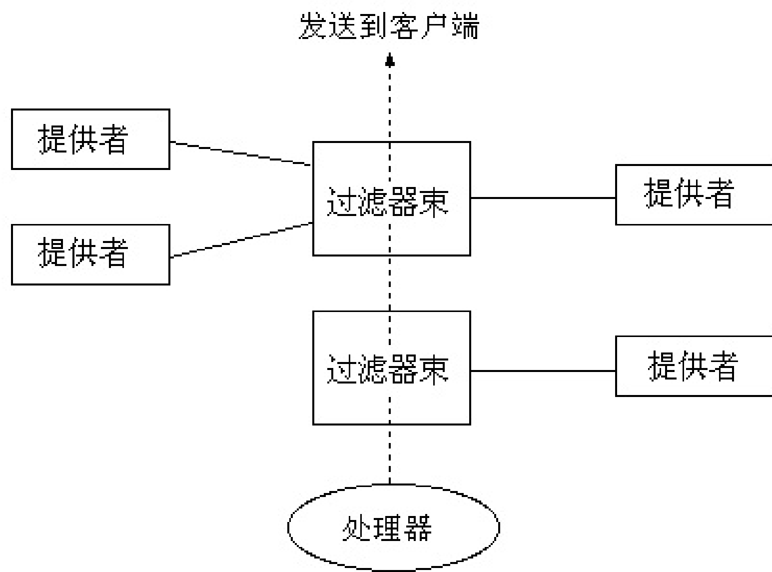
`mod_filter` by contrast gives server administrators a great deal of flexibility in configuring the filter chain. In fact, filters can be inserted based on any Request Header, Response Header or Environment Variable. This generalises the limited flexibility offered by `AddOutputFilterByType` , and fixes it to work correctly with dynamic content, regardless of the content generator. The ability to dispatch based on Environment Variables offers the full flexibility of configuration with `mod_rewrite` to anyone who needs it.

Filter Declarations, Providers and Chains



<dfn class="calibre50">Figure 1:</dfn> The traditional filter model

In the traditional model, output filters are a simple chain from the content generator (handler) to the client. This works well provided the filter chain can be correctly configured, but presents problems when the filters need to be configured dynamically based on the outcome of the handler.



<dfn class="calibre50">Figure 2:</dfn> The `mod_filter` model

`mod_filter` works by introducing indirection into the filter chain. Instead of inserting filters in the chain, we insert a filter harness which in turn dispatches conditionally to a filter provider. Any content filter may be used as a provider to `mod_filter`; no change to existing filter modules is required (although it may be possible to simplify them). There can be multiple providers for one filter, but no more than one provider will run for any single request.

A filter chain comprises any number of instances of the filter harness, each of which may have any number of providers. A special case is that of a single provider with unconditional dispatch: this is equivalent to inserting the provider filter directly into the chain.

Configuring the Chain

There are three stages to configuring a filter chain with `mod_filter`. For details of the directives, see below.

Declare Filters

The `FilterDeclare` directive declares a filter, assigning it a name and filter type. Required only if the filter is not the default type AP_FTYPE_RESOURCE.

Register Providers

The `FilterProvider` directive registers a provider with a filter. The filter may have been declared with `FilterDeclare`; if not, FilterProvider will implicitly declare it with the default type AP_FTYPE_RESOURCE. The provider must have been registered with `ap_register_output_filter` by some module. The remaining arguments to `FilterProvider` are a dispatch criterion and a match string. The former may be an HTTP request or response header, an environment variable, or the Handler used by this request. The latter is matched to it for each request, to determine whether this provider will be used to implement the filter for this request.

Configure the Chain

The above directives build components of a smart filter chain, but do not configure it to run. The `FilterChain` directive builds a filter chain from smart filters declared, offering the flexibility to insert filters at the beginning or end of the chain, remove a filter, or clear the chain.

Examples

Server side Includes (SSI)

A simple case of using `mod_filter` in place of `AddOutputFilterByType`

```
FilterDeclare SSI

FilterProvider SSI INCLUDES resp=Content-Type $text/html

FilterChain SSI
```

Server side Includes (SSI)

The same as the above but dispatching on handler (classic SSI behaviour; .shtml files get processed).

```
FilterProvider SSI INCLUDES Handler server-parsed

FilterChain SSI
```

Emulating mod_gzip with mod_deflate

Insert INFLATE filter only if "gzip" is NOT in the Accept-Encoding header. This filter runs with ftype CONTENT_SET.

```
FilterDeclare gzip CONTENT_SET

FilterProvider gzip inflate req=Accept-Encoding !$gzip

FilterChain gzip
```

Image Downsampling

Suppose we want to downsample all web images, and have filters for GIF, JPEG and PNG.

```
      FilterProvider unpack jpeg_unpack Content-Type $image/jpeg

      FilterProvider unpack gif_unpack Content-Type $image/gif

      FilterProvider unpack png_unpack Content-Type $image/png

      FilterProvider downsample downsample_filter Content-Type $image

      FilterProtocol downsample "change=yes"

      FilterProvider repack jpeg_pack Content-Type $image/jpeg

      FilterProvider repack gif_pack Content-Type $image/gif

      FilterProvider repack png_pack Content-Type $image/png

      &lt;Location /image-filter&gt;

 FilterChain unpack downsample repack
      &lt;/Location&gt;
```

Protocol Handling

Historically, each filter is responsible for ensuring that whatever changes it makes are correctly represented in the HTTP response headers, and that it does not run when it would make an illegal change. This imposes a burden on filter authors to re-implement some common functionality in every filter:

- Many filters will change the content, invalidating existing content tags, checksums, hashes, and lengths.

- Filters that require an entire, unbroken response in input need to ensure they don't get byteranges from a backend.
- Filters that transform output in a filter need to ensure they don't violate a `Cache-Control: no-transform` header from the backend.
- Filters may make responses uncacheable.

`mod_filter` aims to offer generic handling of these details of filter implementation, reducing the complexity required of content filter modules. This is work-in-progress; the `FilterProtocol` implements some of this functionality for back-compatibility with Apache 2.0 modules. For httpd 2.1 and later, the `ap_register_output_filter_protocol` 和 `ap_filter_protocol` API enables filter modules to declare their own behaviour.

At the same time, `mod_filter` should not interfere with a filter that wants to handle all aspects of the protocol. By default (i.e. in the absence of any `FilterProtocol` directives), `mod_filter` will leave the headers untouched.

At the time of writing, this feature is largely untested, as modules in common use are designed to work with 2.0. Modules using it should test it carefully.

FilterChain 指令

说明	Configure the filter chain
语法	`FilterChain [+=-@!]filter-name ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Options
状态	基本(B)
模块	mod_filter

This configures an actual filter chain, from declared filters. `FilterChain` takes any number of arguments, each optionally preceded with a single-character control that determines what to do:

`+filter-name`

Add filter-name to the end of the filter chain

`@filter-name`

Insert filter-name at the start of the filter chain

`-filter-name`

Remove filter-name from the filter chain

`=filter-name`

Empty the filter chain and insert filter-name

`!`

Empty the filter chain

`filter-name`

Equivalent to `+filter-name`

FilterDeclare 指令

说明	Declare a smart filter
语法	`FilterDeclare filter-name [type]`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Options
状态	基本(B)
模块	mod_filter

This directive declares an output filter together with a header or environment variable that will determine runtime configuration. The first argument is a filter-name for use in `FilterProvider` , `FilterChain` 和 `FilterProtocol` directives.

The final (optional) argument is the type of filter, and takes values of `ap_filter_type` - namely `RESOURCE` (the default), `CONTENT_SET` , `PROTOCOL` , `TRANSCODE` , `CONNECTION` 或 `NETWORK` .

FilterProtocol 指令

说明	Deal with correct HTTP protocol handling
语法	`FilterProtocol filter-name [provider-name] proto-flags`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Options
状态	基本(B)
模块	mod_filter

This directs `mod_filter` to deal with ensuring the filter doesn't run when it shouldn't, and that the HTTP response headers are correctly set taking into account the effects of the filter.

There are two forms of this directive. With three arguments, it applies specifically to a filter-name and a provider-name for that filter. With two arguments it applies to a filter-name whenever the filter runs *any* provider.

proto-flags is one or more of

`change=yes`

The filter changes the content, including possibly the content length

`change=1:1`

The filter changes the content, but will not change the content length

`byteranges=no`

The filter cannot work on byteranges and requires complete input

`proxy=no`

The filter should not run in a proxy context

`proxy=transform`

The filter transforms the response in a manner incompatible with the HTTP `Cache-Control: no-transform` header.

`cache=no`

The filter renders the output uncacheable (eg by introducing randomised content changes)

FilterProvider 指令

说明	**Register a content filter**
语法	`FilterProvider filter-name provider-name [req|resp|env]=dispatch match`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Options
状态	基本(B)
模块	mod_filter

This directive registers a *provider* for the smart filter. The provider will be called if and only if the match declared here matches the value of the header or environment variable declared as dispatch.

provider-name must have been registered by loading a module that registers the name with `ap_register_output_filter` .

dispatch argument is a string with optional `req=` , `resp=` 或 `env=` prefix causing it to dispatch on (respectively) the request header, response header, or environment variable named. In the absence of a prefix, it defaults to a response header. A special case is the word `handler` , which causes `mod_filter` to dispatch on the content handler.

match argument specifies a match that will be applied to the filter's dispatch criterion. The match may be a string match (exact match or substring), a regex, an integer (greater, lessthan or equals), or unconditional. The first characters of the match argument determines this:

First, if the first character is an exclamation mark (`!`), this reverses the rule, so the provider will be used if and only if the match *fails*.

Second, it interprets the first character excluding any leading `!` as follows:

Character	Description
(none)	exact match
`$`	substring match
`/`	regex match (delimited by a second `/`)
`=`	integer equality
<	integer less-than
<=	integer less-than or equal
>	integer greater-than
>=	integer greater-than or equal
`*`	Unconditional match

FilterTrace 指令

说明	Get debug/diagnostic information from `mod_filter`
语法	`FilterTrace filter-name level`
作用域	server config, virtual host, directory
状态	基本(B)
模块	mod_filter

This directive generates debug information from `mod_filter`. It is designed to help test and debug providers (filter modules), although it may also help with `mod_filter` itself.

The debug output depends on the level set:

`0` (default)

No debug information is generated.

`1`

`mod_filter` will record buckets and brigades passing through the filter to the error log, before the provider has processed them. This is similar to the information generated by mod_diagnostics.

`2` (not yet implemented)

Will dump the full data passing through to a tempfile before the provider. **For single-user debug only**; this will not support concurrent hits.

Apache模块 mod_headers

说明	允许通过配置文件控制任意的HTTP请求和应答头信息
状态	扩展(E)
模块名	headers_module
源文件	mod_headers.c
兼容性	`RequestHeader` 仅在 Apache 2.0 中有效

概述

这个模块提供了一些指令用于控制和修改HTTP请求头和应答头。这些头可以被合并、替换、删除。

处理顺序

由 `mod_headers` 提供的指令几乎可以出现在配置文件的任何部分。并可以封装在配置段中以限制其作用范围。

指令的处理顺序很重要，它取决于指令本身在配置文件中的位置和所属配置段在配置文件中的位置。下面的指令如果颠倒一下顺序将会导致完全不同的结果：

```
RequestHeader append MirrorID "mirror 12"
RequestHeader unset MirrorID
```

当前顺序下， `MirrorID` 头不会被设置。若颠倒一下顺序， `MirrorID` 头将被设为"mirror 12"。

前处理和后处理

`mod_headers` 可以应用在请求被处理之前或之后。通常的模式是"后处理"，也就是在请求处理完毕之后、发送应答之前设置应答头。[原文：when Request Headers are set immediately before running the content generator and Response Headers just as the response is sent down the wire.]在实际工作的服务器上应当始终使用"后处理"模式。

"前处理"模式应当仅仅作为开发者使用的一种测试/调试辅助工具。可以在指令中使用 `early` 关键字启用"前处理"模式，此时将在处理请求之前设置请求头。这样就可以模拟各种不同的请求以协助调试。

由于"前处理"模式的指令在将URL映射到文件系统之前就生效了，不能依赖于所请求的路径。所以"前处理"模式的指令只能用在主服务器和虚拟主机部分的配置中，而不能用于 `<Directory>` 或 `<Location>` 配置段中。

示例

1. 将所有以"TS"开头的请求头复制到应答头中：

```
Header echo ^TS
```

2. 在应答中添加一个 `MyHeader` 头来包含服务端接受到请求的时间戳和经过多少时间以后才完成对该请求的处理并作出应答。这个头可以让客户端知道瓶颈位于服务端还是位于服务端和客户端之间的线路。

```
Header add MyHeader "%D %t"
```

上面的设置将会添加如下应答头内容：

```
MyHeader: D=3775428 t=991424704447256
```

3. 向Joe问好(Hello)：

```
Header add MyHeader "Hello Joe. It took %D microseconds \
for Apache to serve this request."
```

上面的设置将会添加如下应答头内容：

```
MyHeader: Hello Joe. It took D=3775428 microseconds for Apache to serve thi
```

4. 当且仅当"MyRequestHeader"出现在请求头中的时候才在应答中发送" `MyHeader` "头。这个对根据特定的客户端构造特定的应答头很有用。注意，下面的例子需要 `mod_setenvif` 模块的支持。

```
SetEnvIf MyRequestHeader value HAVE_MyRequestHeader
Header add MyHeader "%D %t mytext" env=HAVE_MyRequestHeader
```

如果请求中出现" `MyRequestHeader: value` "头，应答中将会包含下面的头：

```
MyHeader: D=3775428 t=991424704447256 mytext
```

Header 指令

说明	配置HTTP应答头
语法	`Header [condition] set|append|add|unset|echo header [value] [ea`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_headers

这个指令可以替换、合并、删除HTTP应答头。应答头紧跟在内容处理器和输出过滤器完工之后生成，这时候才能对头进行修改。

condition选项可以是 `onsuccess` 或 `always` 。它决定了将对哪个内部头(internal header)表进行操作。 `onsuccess` 表示" `2xx` "状态码， `always` 表示所有状态码(包含" `2xx` ")。特别地，如果你想撤销由某个模块设置的头，你应当做做试验，以找到是那个表产生的影响。

该指令执行的动作是由第二个参数决定的。这个参数取值范围如下：

`set`

设置新的或修改已经存在的同名应答头。value可以是一个格式字符串。

`append`

向同名应答头添加新内容而不修改原来已经存在的旧内容。当向一个已经存在的头添加新值时，将用逗号与原来已经存在的旧值分开。这是向HTTP头赋以多个值的标准方法。

`add`

向应答中添加新的头而不修改原来已经存在的头(即使同名)。这将可能导致有两个或更多的应答头具有相同的名字，从而导致意想不到的后果，所以通常不使用这种方法而用 `append` 来代替它。

`unset`

去除应答中同名的头(若存在的话)。如果有多个头同名，则会被全部去除。value必须被省略。

`echo`

将请求中同名的头复制到应答中。header可以是一个正则表达式。value必须被省略。

这个参数后面必须要跟一个header名字(结尾的冒号可要可不要)。 `set` , `append` , `add` , `unset` 是大小写无关的。用于 `echo` 的header是大小写敏感的，并且可以是一个正则表达式。

对于 `add` , `append` , `set` 来说，value是第三个参数。如果value包含空格则必须用双引号(")括起来。value可以是一个普通字符串或包含格式说明符的字符串，value支持下列格式字符串：

格式	描述
`%%`	百分号(%)
`%t`	接收到请求的微秒时间戳(相对于1970-1-1 00:00:00 UCT)，外加一个" `t=` "前缀。
`%D`	从接收到请求到完成对该请求的处理并作出应答共花费了多少微秒，外加一个" `D=` "前缀。
`%{FOOBAR}e`	环境变量 `FOOBAR` 的内容
`%{FOOBAR}s`	SSL环境变量 `FOOBAR` 的内容(如果启用了 `mod_ssl`)

注意

" `%s` "格式符仅在Apache 2.1及以后的版本中可用。它可以代替" `%e` "以避免" `SSLOptions +StdEnvVars` "带来的额外开销。如果因为别的原因必须开启" `SSLOptions +StdEnvVars` "，那么" `%e` "将比" `%s` "更加合适。

`Header` 后面可以跟一个用于指定生效条件的额外参数(或者用 `early` 表示"前处理")。如果在" `env=...` "参数中指定的环境变量存在(或用" `env=!...` "表示不存在)，那么 `Header` 指令指定的动作将会生效，否则将不会生效。

除非使用early模式，否则 `Header` 指令将在应答最后被发送到网络前处理。这意味着可以设置和改写绝大多数应答头，除了自己添加的应答头。

RequestHeader 指令

说明	配置HTTP请求头
语法	`RequestHeader set|append|add|unset header [value] [early|env=[`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_headers
兼容性	仅在 Apache 2.0 中有效

这个指令可以替换、合并、删除HTTP应答头。请求头将会在内容处理器运行之前被修改。修改的动作由第一个参数决定，其取值范围如下：

`set`

设置新的或修改已经存在的同名请求头。

`append`

向同名请求头添加新内容而不修改原来已经存在的旧内容。当向一个已经存在的头添加新值时，将用逗号与原来已经存在的旧值分开。这是向HTTP头赋以多个值的标准方法。

`add`

向请求中添加新的头而不修改原来已经存在的头(即使同名)。这将可能导致有两个或更多的请求头具有相同的名字，从而导致意想不到的后果，所以通常不使用这种方法而用 `append` 来代替它。

`unset`

去除请求中同名的头(若存在的话)。如果有多个头同名，则会被全部去除。value必须被省略。

这些参数后面必须要跟一个header名(结尾的冒号可要可不要，且大小写无关)。对于 `add` , `append` , `set` 来说，value是第三个参数。如果value包含空格则必须用双引号(")括起来。对于 `unset` 来说则不需要value参数。value可以是一个普通字符串或包含格式说明符的字符串，格式字符串的用法与 `Header` 指令一样。

`RequestHeader` 后面可以跟一个用于指定生效条件的额外参数(或者用 `early` 表示"前处理")。如果在" `env=...` "参数中指定的环境变量存在(或用" `env=!...` "表示不存在)，那么 `RequestHeader` 指令指定的动作将会生效，否则将不会生效。

除非使用early模式，否则 `RequestHeader` 将在请求被处理之前生效。这样，由浏览器和Apache输入过滤器产生的请求头都可以被该指令处理。

Apache模块 mod_ident

说明	实现**RFC1413**规定的**ident**查找
状态	扩展(E)
模块名	ident_module
源文件	mod_ident.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

该模块只有在远程主机上运行了RFC 1413兼容的守护进程的情况下才能查询连接者的身份。

IdentityCheck 指令

说明	启用对远端用户的**RFC1413**身份鉴定的日志
语法	`IdentityCheck On|Off`
默认值	`IdentityCheck Off`
作用域	server config, virtual host, directory
状态	扩展(E)
模块	mod_ident
兼容性	Apache 2.1以后从服务器核心中移出到该模块

当客户端运行了identd或类似程序时，此指令将决定是否为每个链接记录RFC 1413兼容的远端用户名。这些信息将记入访问日志中(使用" `%l` "格式字符串)。

此信息除了用于基本的跟踪以外是不应该多加信任的。

因为对每个请求都会进行这样的查询，所以这样设置可能会使您的服务器访问出现严重的延迟。当涉及到防火墙或代理的时候，每次查询都会因为可能的失败而对每次点击增加 `IdentityCheckTimeout` 秒的延迟。所以一般来说对于公网上的服务器这个指令并不是很有用。

IdentityCheckTimeout 指令

说明	设置**ident**请求超时
语法	`IdentityCheckTimeout seconds`
默认值	`IdentityCheckTimeout 30`
作用域	server config, virtual host, directory
状态	扩展(E)
模块	mod_ident

该指令设置了ident请求的超时时间。默认值"30"(秒)也是RFC 1413的推荐值，这主要是考虑到可能的网络延时。当然，你也可以根据你自己的网络状况进行调整。

Apache模块 mod_imagemap

说明	处理服务器端图像映射
状态	基本(B)
模块名	imagemap_module
源文件	mod_imagemap.c

概述

This module processes `.map` files, thereby replacing the functionality of the `imagemap` CGI program. Any directory or document type configured to use the handler `imap-file` (using either `AddHandler` 或 `SetHandler`) will be processed by this module.

The following directive will activate files ending with `.map` as imagemap files:

```
AddHandler imap-file map
```

Note that the following is still supported:

```
AddType application/x-httpd-imap map
```

However, we are trying to phase out "magic MIME types" so we are deprecating this method.

New Features

The imagemap module adds some new features that were not possible with previously distributed imagemap programs.

- URL references relative to the Referer: information.
- Default `<base>` assignment through a new map directive `base` .
- No need for `imagemap.conf` file.
- Point references.
- Configurable generation of imagemap menus.

Imagemap File

The lines in the imagemap files can have one of several formats:

```
directive value [<var class="calibre40">x</var>, <var class="calibre40">y</var> ...

directive value "<var class="calibre40">Menu text</var>" [<var class="calibre40">x<
...]

directive value <var class="calibre40">x</var>, <var class="calibre40">y</var> ...
```

The directive is one of `base` , `default` , `poly` , `circle` , `rect` , or `point` . The value is an absolute or relative URL, or one of the special values listed below. The coordinates are `x, y` pairs separated by whitespace. The quoted text is used as the text of the link if a imagemap menu is generated. Lines beginning with '#' are comments.

Imagemap File Directives

There are six directives allowed in the imagemap file. The directives can come in any order, but are processed in the order they are found in the imagemap file.

`base` Directive

Has the effect of `<base href="value">` . The non-absolute URLs of the map-file are taken relative to this value. The `base` directive overrides `ImapBase` as set in a `.htaccess` file or in the server configuration files. In the absence of an `ImapBase` configuration directive, `base` defaults to `http://server_name/` .

`base_uri` is synonymous with `base` . Note that a trailing slash on the URL is significant.

`default` Directive

The action taken if the coordinates given do not fit any of the `poly` , `circle` 或 `rect` directives, and there are no `point` directives. Defaults to `nocontent` in the absence of an `ImapDefault` configuration setting, causing a status code of `204 No Content` to be returned. The client should keep the same page displayed.

`poly` Directive

Takes three to one-hundred points, and is obeyed if the user selected coordinates fall within the polygon defined by these points.

`circle`

Takes the center coordinates of a circle and a point on the circle. Is obeyed if the user selected point is with the circle.

`rect` Directive

Takes the coordinates of two opposing corners of a rectangle. Obeyed if the point selected is within this rectangle.

`point` Directive

Takes a single point. The point directive closest to the user selected point is obeyed if no other directives are satisfied. Note that `default` will not be followed if a `point` directive is present and valid coordinates are given.

Values

The values for each of the directives can any of the following:

a URL

The URL can be relative or absolute URL. Relative URLs can contain '..' syntax and will be resolved relative to the `base` value.

`base` itself will not resolved according to the current value. A statement `base mailto:` will work properly, though.

`map`

Equivalent to the URL of the imagemap file itself. No coordinates are sent with this, so a menu will be generated unless `ImapMenu` is set to `none`.

`menu`

Synonymous with `map`.

`referer`

Equivalent to the URL of the referring document. Defaults to `http://servername/` if no `Referer:` header was present.

`nocontent`

Sends a status code of `204 No Content`, telling the client to keep the same page displayed. Valid for all but `base`.

`error`

Fails with a `500 Server Error`. Valid for all but `base`, but sort of silly for anything but `default`.

Coordinates

`0,0 200,200`

A coordinate consists of an x and a y value separated by a comma. The coordinates are separated from each other by whitespace. To accommodate the way Lynx handles imagemaps, should a user select the coordinate `0,0`, it is as if no coordinate had been

selected.

Quoted Text

`"Menu Text"`

After the value or after the coordinates, the line optionally may contain text within double quotes. This string is used as the text for the link if a menu is generated:

```
&lt;a href="http://foo.com/"&gt;Menu text&lt;/a&gt;
```

If no quoted text is present, the name of the link will be used as the text:

```
&lt;a href="http://foo.com/"&gt;http://foo.com&lt;/a&gt;
```

If you want to use double quotes within this text, you have to write them as `"`.

Example Mapfile

```
#Comments are printed in a 'formatted' or 'semiformatted' menu.

#And can contain html tags. <hr>

base referer

poly map "Could I have a menu, please?" 0,0 0,10 10,10 10,0

rect .. 0,0 77,27 "the directory of the referer"

circle http://www.inetnebr.com/lincoln/feedback/ 195,0 305,27

rect another_file "in same directory as referer" 306,0 419,27

point http://www.zyzzyva.com/ 100,100

point http://www.tripod.com/ 200,200

rect mailto:nate@tripod.com 100,150 200,0 "Bugs?"
```

Referencing your mapfile

HTML example

```
     <a href="/maps/imagemap1.map">

<img ismap src="/images/imagemap1.gif">
     </a>
```

XHTML example

```
      <a href="/maps/imagemap1.map">

 <img ismap="ismap" src="/images/imagemap1.gif" />
      </a>
```

ImapBase 指令

说明	**Default `base` for imagemap files**
语法	`ImapBase map|referer|URL`
默认值	`ImapBase http://servername/`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_imagemap

`ImapBase` directive sets the default `base` used in the imagemap files. Its value is overridden by a `base` directive within the imagemap file. If not present, the `base` defaults to `http://servername/` .

参见

- `UseCanonicalName`

ImapDefault 指令

说明	**Default action when an imagemap is called with coordinates that are not explicitly mapped**
语法	`ImapDefault error|nocontent|map|referer|URL`
默认值	`ImapDefault nocontent`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_imagemap

`ImapDefault` directive sets the default `default` used in the imagemap files. Its value is overridden by a `default` directive within the imagemap file. If not present, the `default` action is `nocontent`, which means that a `204 No Content` is sent to the client. In this case, the client should continue to display the original page.

ImapMenu 指令

说明	Action if no coordinates are given when calling an imagemap
语法	`ImapMenu none|formatted|semiformatted|unformatted`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Indexes
状态	基本(B)
模块	mod_imagemap

`ImapMenu` directive determines the action taken if an imagemap file is called without valid coordinates.

`none`

If ImapMenu is `none`, no menu is generated, and the `default` action is performed.

`formatted`

A `formatted` menu is the simplest menu. Comments in the imagemap file are ignored. A level one header is printed, then an hrule, then the links each on a separate line. The menu has a consistent, plain look close to that of a directory listing.

`semiformatted`

In the `semiformatted` menu, comments are printed where they occur in the imagemap file. Blank lines are turned into HTML breaks. No header or hrule is printed, but otherwise the menu is the same as a `formatted` menu.

`unformatted`

Comments are printed, blank lines are ignored. Nothing is printed that does not appear in the imagemap file. All breaks and headers must be included as comments in the imagemap file. This gives you the most flexibility over the appearance of your menus, but requires you to treat your map files as HTML instead of plaintext.

Apache模块 mod_include

说明	实现服务端包含文档(SSI)处理
状态	基本(B)
模块名	include_module
源文件	mod_include.c
兼容性	Implemented as an output filter since Apache 2.0

概述

This module provides a filter which will process files before they are sent to the client. The processing is controlled by specially formatted SGML comments, referred to as <dfn class="calibre27">elements</dfn>. These elements allow conditional text, the inclusion of other files or programs, as well as the setting and printing of environment variables.

Enabling Server-Side Includes

Server Side Includes are implemented by the `INCLUDES` filter. If documents containing server-side include directives are given the extension .shtml, the following directives will make Apache parse them and assign the resulting document the mime type of `text/html` :

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

The following directive must be given for the directories containing the shtml files (typically in a `<Directory>` section, but this directive is also valid in `.htaccess` files if `AllowOverride Options` is set):

```
Options +Includes
```

For backwards compatibility, the `server-parsed` 处理器 also activates the INCLUDES filter. As well, Apache will activate the INCLUDES filter for any document with mime type `text/x-server-parsed-html` 或 `text/x-server-parsed-html3` (and the resulting output will have the mime type `text/html`).

For more information, see our Tutorial on Server Side Includes.

PATH_INFO with Server Side Includes

Files processed for server-side includes no longer accept requests with `PATH_INFO` (trailing pathname information) by default. You can use the `AcceptPathInfo` directive to configure the server to accept requests with `PATH_INFO`.

Basic Elements

The document is parsed as an HTML document, with special commands embedded as SGML comments. A command has the syntax:

```
     <!--#<var class="calibre40">element</var> <var class="calibre40">attribute</var>=<v
<var class="calibre40">attribute</var>=<var class="calibre40">value</var> ... -->
```

The value will often be enclosed in double quotes, but single quotes (`'`) and backticks (`` ` ``) are also possible. Many commands only allow a single attribute-value pair. Note that the comment terminator (-->) should be preceded by whitespace to ensure that it isn't considered part of an SSI token. Note that the leading <!--# is *one* token and may not contain any whitespaces.

The allowed elements are listed in the following table:

Element	Description
`config`	configure output formats
`echo`	print variables
`exec`	execute external programs
`fsize`	print size of a file
`flastmod`	print last modification time of a file
`include`	include a file
`printenv`	print all available variables
`set`	set a value of a variable

SSI elements may be defined by modules other than `mod_include`. In fact, the `exec` element is provided by `mod_cgi`, and will only be available if this module is loaded.

The config Element

This command controls various aspects of the parsing. The valid attributes are:

`echomsg` (*Apache 2.1 and later*)

The value is a message that is sent back to the client if the `echo` element attempts to echo an undefined variable. This overrides any `SSIUndefinedEcho` directives.

`errmsg`

The value is a message that is sent back to the client if an error occurs while parsing the document. This overrides any `SSIErrorMsg` directives.

`sizefmt`

The value sets the format to be used which displaying the size of a file. Valid values are `bytes` for a count in bytes, or `abbrev` for a count in Kb or Mb as appropriate, for example a size of 1024 bytes will be printed as "1K".

`timefmt`

The value is a string to be used by the `strftime(3)` library routine when printing dates.

The echo Element

This command prints one of the include variables, defined below. If the variable is unset, the result is determined by the `SSIUndefinedEcho` directive. Any dates printed are subject to the currently configured `timefmt`.

Attributes:

`var`

The value is the name of the variable to print.

`encoding`

Specifies how Apache should encode special characters contained in the variable before outputting them. If set to `none`, no encoding will be done. If set to `url`, then URL encoding (also known as %-encoding; this is appropriate for use within URLs in links, etc.) will be performed. At the start of an `echo` element, the default is set to `entity`, resulting in entity encoding (which is appropriate in the context of a block-level HTML element, 例如， a paragraph of text). This can be changed by adding an `encoding` attribute, which will remain in effect until the next `encoding` attribute is encountered or the element ends, whichever comes first.

`encoding` attribute must *precede* the corresponding `var` attribute to be effective, and only special characters as defined in the ISO-8859-1 character encoding will be encoded. This encoding process may not have the desired result if a different character encoding is in use.

In order to avoid cross-site scripting issues, you should *always* encode user supplied data.

The exec Element

`exec` command executes a given shell command or CGI script. It requires `mod_cgi` to be present in the server. If `Options` `IncludesNOEXEC` is set, this command is completely disabled. The valid attributes are:

`cgi`

The value specifies a (%-encoded) URL-path to the CGI script. If the path does not begin with a slash (/), then it is taken to be relative to the current document. The document referenced by this path is invoked as a CGI script, even if the server would not normally recognize it as such. However, the directory containing the script must be enabled for CGI scripts (with `ScriptAlias` 或 `Options` `ExecCGI`).

The CGI script is given the `PATH_INFO` and query string (`QUERY_STRING`) of the original request from the client; these *cannot* be specified in the URL path. The include variables will be available to the script in addition to the standard CGI environment.

示例

```
&lt;!--#exec cgi="/cgi-bin/example.cgi" --&gt;
```

If the script returns a `Location:` header instead of output, then this will be translated into an HTML anchor.

`include virtual` element should be used in preference to `exec cgi` . In particular, if you need to pass additional arguments to a CGI program, using the query string, this cannot be done with `exec cgi` , but can be done with `include virtual` , as shown here:

```
&lt;!--#include virtual="/cgi-bin/example.cgi?argument=value" --&gt;
```

`cmd`

The server will execute the given string using `/bin/sh` . The include variables are available to the command, in addition to the usual set of CGI variables.

The use of `#include virtual` is almost always prefered to using either `#exec cgi` 或 `#exec cmd` . The former (`#include virtual`) uses the standard Apache sub-request mechanism to include files or scripts. It is much better tested and maintained.

In addition, on some platforms, like Win32, and on unix when using suexec, you cannot pass arguments to a command in an `exec` directive, or otherwise include spaces in the command. Thus, while the following will work under a non-suexec configuration on unix, it will not produce the desired result under Win32, or when running suexec:

```
        &lt;!--#exec cmd="perl /path/to/perlscript arg1 arg2" --&gt;
```

The fsize Element

This command prints the size of the specified file, subject to the `sizefmt` format specification. Attributes:

`file`

The value is a path relative to the directory containing the current document being parsed.

`virtual`

The value is a (%-encoded) URL-path. If it does not begin with a slash (/) then it is taken to be relative to the current document. Note, that this does *not* print the size of any CGI output, but the size of the CGI script itself.

The flastmod Element

This command prints the last modification date of the specified file, subject to the `timefmt` format specification. The attributes are the same as for the `fsize` command.

The include Element

This command inserts the text of another document or file into the parsed file. Any included file is subject to the usual access control. If the directory containing the parsed file has Options `IncludesNOEXEC` set, then only documents with a text MIME-type (`text/plain` , `text/html` etc.) will be included. Otherwise CGI scripts are invoked as normal using the complete URL given in the command, including any query string.

An attribute defines the location of the document; the inclusion is done for each attribute given to the include command. The valid attributes are:

`file`

The value is a path relative to the directory containing the current document being parsed. It cannot contain `../` , nor can it be an absolute path. Therefore, you cannot include files that are outside of the document root, or above the current document in the directory structure. The `virtual` attribute should always be used in preference to this one.

```
<a id="calibre_link-569" name="includevirtual" class="pcalibre1 pcalibre"&gt;virtual&lt;/a
```

The value is a (%-encoded) URL-path. The URL cannot contain a scheme or hostname, only a path and an optional query string. If it does not begin with a slash (/) then it is taken to be relative to the current document.

A URL is constructed from the attribute, and the output the server would return if the URL were accessed by the client is included in the parsed output. Thus included files can be nested.

If the specified URL is a CGI program, the program will be executed and its output inserted in place of the directive in the parsed file. You may include a query string in a CGI url:

```
&lt;!--#include virtual="/cgi-bin/example.cgi?argument=value" --&gt;
```

`include virtual` should be used in preference to `exec cgi` to include the output of CGI programs into an HTML document.

The printenv Element

This prints out a listing of all existing variables and their values. Special characters are entity encoded (see the `echo` element for details) before being output. There are no attributes.

示例

```
<!--#printenv -->
```

The set Element

This sets the value of a variable. Attributes:

`var`

The name of the variable to set.

`value`

The value to give a variable.

示例

```
<!--#set var="category" value="help" -->
```

Include Variables

In addition to the variables in the standard CGI environment, these are available for the `echo` command, for `if` 和 `elif`, and to any program invoked by the document.

`DATE_GMT`

The current date in Greenwich Mean Time.

`DATE_LOCAL`

The current date in the local time zone.

`DOCUMENT_NAME`

The filename (excluding directories) of the document requested by the user.

`DOCUMENT_URI`

The (%-decoded) URL path of the document requested by the user. Note that in the case of nested include files, this is *not* the URL for the current document.

`LAST_MODIFIED`

The last modification date of the document requested by the user.

`QUERY_STRING_UNESCAPED`

If a query string is present, this variable contains the (%-decoded) query string, which is *escaped* for shell usage (special characters like `&` etc. are preceded by backslashes).

Variable Substitution

Variable substitution is done within quoted strings in most cases where they may reasonably occur as an argument to an SSI directive. This includes the `config` , `exec` , `flastmod` , `fsize` , `include` , `echo` , and `set` directives, as well as the arguments to conditional operators. You can insert a literal dollar sign into the string using backslash quoting:

```
<!--#if expr="$a = \$test" -->
```

If a variable reference needs to be substituted in the middle of a character sequence that might otherwise be considered a valid identifier in its own right, it can be disambiguated by enclosing the reference in braces, *a la* shell substitution:

```
<!--#set var="Zed" value="${REMOTE_HOST}_${REQUEST_METHOD}" -->
```

This will result in the `Zed` variable being set to " `X_Y` " if `REMOTE_HOST` is " `X` " and `REQUEST_METHOD` is " `Y` ".

The below example will print "in foo" if the `DOCUMENT_URI` is `/foo/file.html` , "in bar" if it is `/bar/file.html` and "in neither" otherwise:

```
      <!--#if expr='"$DOCUMENT_URI" = "/foo/file.html"' -->

in foo
      <!--#elif expr='"$DOCUMENT_URI" = "/bar/file.html"' -->

in bar
      <!--#else -->

in neither
      <!--#endif -->
```

Flow Control Elements

The basic flow control elements are:

```
      <!--#if expr="<var class="calibre40">test_condition</var>" -->

      <!--#elif expr="<var class="calibre40">test_condition</var>" -->

      <!--#else -->

      <!--#endif -->
```

`if` element works like an if statement in a programming language. The test condition is evaluated and if the result is true, then the text until the next `elif` , `else` 或 `endif` element is included in the output stream.

`elif` 或 `else` statements are be used to put text into the output stream if the original test_condition was false. These elements are optional.

`endif` element ends the `if` element and is required.

test_condition is one of the following:

`string`

true if string is not empty

`string1 = string2 string1 == string2 string1 != string2`

Compare string1 with string2. If string2 has the form `/string2/` then it is treated as a regular expression. Regular expressions are implemented by the PCRE engine and have the same syntax as those in perl 5. Note that `==` is just an alias for `=` and behaves exactly the same way.

If you are matching positive (`=` 或 `==`), you can capture grouped parts of the regular expression. The captured parts are stored in the special variables `$1` .. `$9` .

示例

```
        &lt;!--#if expr="$QUERY_STRING = /^sid=([a-zA-Z0-9]+)/" --&gt;

  &lt;!--#set var="session" value="$1" --&gt;
        &lt;!--#endif --&gt;
```

```
string1 &lt; string2 string1 &lt;= string2 string1 &gt; string2 string1 &gt;= string2
```

Compare string1 with string2. Note, that strings are compared *literally* (using `strcmp(3)`). Therefore the string "100" is less than "20".

`( test_condition )`

true if test_condition is true

`! test_condition`

true if test_condition is false

`test_condition1 && test_condition2`

true if both test_condition1和test_condition2 are true

`test_condition1 || test_condition2`

true if either test_condition1或test_condition2 is true

" `=` " and " `!=` " bind more tightly than " `&&` " and " `||` ". " `!` " binds most tightly. Thus, the following are equivalent:

```
      <!--#if expr="$a = test1 && $b = test2" -->

      <!--#if expr="($a = test1) && ($b = test2)" -->
```

The boolean operators `&&` 和 `||` share the same priority. So if you want to bind such an operator more tightly, you should use parentheses.

Anything that's not recognized as a variable or an operator is treated as a string. Strings can also be quoted: `'string'` . Unquoted strings can't contain whitespace (blanks and tabs) because it is used to separate tokens such as variables. If multiple strings are found in a row, they are concatenated using blanks. So,

```
<var class="calibre40">string1</var>    <var class="calibre40">string2</var> results in <

和

'<var class="calibre40">string1</var>    <var class="calibre40">string2</var>' results in
```

Optimization of Boolean Expressions

If the expressions become more complex and slow down processing significantly, you can try to optimize them according to the evaluation rules:

- Expressions are evaluated from left to right
- Binary boolean operators (`&&` 和 `||`) are short circuited wherever possible. In conclusion with the rule above that means, `mod_include` evaluates at first the left expression. If the left result is sufficient to determine the end result, processing stops here. Otherwise it evaluates the right side and computes the end result from both left and right results.
- Short circuit evaluation is turned off as long as there are regular expressions to deal with. These must be evaluated to fill in the backreference variables (`$1` .. `$9`).

If you want to look how a particular expression is handled, you can recompile `mod_include` using the `-DDEBUG_INCLUDE` compiler option. This inserts for every parsed expression tokenizer information, the parse tree and how it is evaluated into the output sent to the client.

SSIEndTag 指令

说明	String that ends an include element
语法	`SSIEndTag tag`
默认值	`SSIEndTag "-->"`
作用域	server config, virtual host
状态	基本(B)
模块	mod_include
兼容性	仅在 Apache 2.0.30 及以后的版本中可用

This directive changes the string that `mod_include` looks for to mark the end of an include element.

示例

```
SSIEndTag "%>"
```

参见

- `SSIStartTag`

SSIErrorMsg 指令

说明	**Error message displayed when there is an SSI error**
语法	`SSIErrorMsg message`
默认值	`SSIErrorMsg "[an error occurred while processing this directive]"`
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	基本(B)
模块	mod_include
兼容性	仅在 Apache 2.0.30 及以后的版本中可用

`SSIErrorMsg` directive changes the error message displayed when `mod_include` encounters an error. For production servers you may consider changing the default error message to `"<!-- Error -->"` so that the message is not presented to the user.

This directive has the same effect as the `<!--#config errmsg=message -->` element.

示例

```
SSIErrorMsg "<!-- Error -->"
```

SSIStartTag 指令

说明	**String that starts an include element**
语法	`SSIStartTag tag`
默认值	`SSIStartTag "<!--#"`
作用域	server config, virtual host
状态	基本(B)
模块	mod_include
兼容性	仅在 Apache 2.0.30 及以后的版本中可用

This directive changes the string that `mod_include` looks for to mark an include element to process.

You may want to use this option if you have 2 servers parsing the output of a file each processing different commands (possibly at different times).

示例

```
SSIStartTag "<%"
SSIEndTag   "%>"
```

The example given above, which also specifies a matching `SSIEndTag`, will allow you to use SSI directives as shown in the example below:

SSI directives with alternate start and end tags

```
<%printenv %>
```

参见

- `SSIEndTag`

SSITimeFormat 指令

说明	Configures the format in which date strings are displayed
语法	`SSITimeFormat formatstring`
默认值	`SSITimeFormat "%A, %d-%b-%Y %H:%M:%S %Z"`
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	基本(B)
模块	mod_include
兼容性	仅在 Apache 2.0.30 及以后的版本中可用

This directive changes the format in which date strings are displayed when echoing `DATE` environment variables. The formatstring is as in `strftime(3)` from the C standard library.

This directive has the same effect as the `<!--#config timefmt=formatstring -->` element.

示例

```
SSITimeFormat "%R, %B %d, %Y"
```

The above directive would cause times to be displayed in the format "22:26, June 14, 2002".

SSIUndefinedEcho 指令

说明	String displayed when an unset variable is echoed
语法	`SSIUndefinedEcho string`
默认值	`SSIUndefinedEcho "(none)"`
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	基本(B)
模块	mod_include
兼容性	仅在 Apache 2.0.34 及以后的版本中可用

This directive changes the string that `mod_include` displays when a variable is not set and "echoed".

示例

```
SSIUndefinedEcho "<!-- undef -->"
```

XBitHack 指令

说明	Parse SSI directives in files with the execute bit set
语法	`XBitHack on|off|full`
默认值	`XBitHack off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Options
状态	基本(B)
模块	mod_include

`XBitHack` directive controls the parsing of ordinary html documents. This directive only affects files associated with the MIME-type `text/html` . `XBitHack` can take on the following values:

`off`

No special treatment of executable files.

`on`

Any `text/html` file that has the user-execute bit set will be treated as a server-parsed html document.

`full`

As for `on` but also test the group-execute bit. If it is set, then set the `Last-modified` date of the returned file to be the last modified time of the file. If it is not set, then no last-modified date is sent. Setting this bit allows clients and proxies to cache the result of the request.

注意

You would not want to use the full option, unless you assure the group-execute bit is unset for every SSI script which might `#include` a CGI or otherwise produces different output on each hit (or could potentially change on subsequent requests).

Apache模块 mod_info

说明	生成Apache配置情况的Web页面
状态	扩展(E)
模块名	info_module
源文件	mod_info.c

概述

要配置 `mod_info` 请将下列内容加入 `httpd.conf` 文件。

```
      <Location /server-info>

SetHandler server-info
      </Location>
```

你可能希望在 `<Location>` 指令中使用 `mod_authz_host` 来限制对服务器配置信息的访问：

```
      <Location /server-info>

SetHandler server-info

        Order deny,allow

        Deny from all

        Allow from yourcompany.com
      </Location>
```

一旦配置完成，你的服务器信息就可以通过访问 `http://your.host.example.com/server-info` 得到。

安全问题

一旦 `mod_info` 被加载，它的处理能力就在所有配置文件中生效，包括 `.htaccess` 文件。这可能给网站带来安全问题。

特别的，该模块还会泄漏许多配置信息，比如：系统路径、用户名/密码、数据库名称等等。而且根据此模块的工作方式，无法对它产生的信息进行屏蔽。因此，应当仅仅在受控环境下使用该模块，并且始终保持警惕。

你可以使用 `mod_authz_host` 来限制对这些敏感信息的访问：

访问控制

```
<Location /server-info>

SetHandler server-info

    Order allow,deny

    # 允许本机自身访问

    Allow from 127.0.0.1

    # 还允许局域网内的另外一台机器访问

    Allow from 192.168.1.17
</Location>
```

选择哪些信息可以被显示

默认情况下显示的信息：所有启用的模块、每个模块的指令说明、每个模块的钩子、当前配置信息。

还可以通过在 `server-info` 后面加上请求字符串来查看特定的信息。比如 `http://your.host.example.com/server-info?config` 将显示所有配置指令。

`?<module-name>`

仅显示与该模块相关的信息

`?config`

仅显示所有配置指令，不按模块分类

`?hooks`

仅显示每个模块所属钩子(Hook)列表

`?list`

仅显示所有启用的模块列表

`?server`

仅显示基本的服务器信息

已知的局限

由于 `mod_info` 提供的信息是根据已经解析过配置树提供的，而不是原始的配置文件，因此有以下局限：

- 立即执行而并不存储的指令不会被列出。包括：`ServerRoot`，`LoadModule`，`LoadFile` 。
- 控制配置文件自身行为的指令不会被列出，包括：`Include`，`<IfModule>`，`<IfDefine>`

。但是通过 `Include` 包含进来的指令将会被列出。
- 配置中的注释不会被列出。
- `.htaccess` 文件中的配置指令不会被列出。
- 容器中的指令按原样列出，但是 `mod_info` 不会计算 `</Directory>` 容器中的行号。
- 第三方模块(如 `mod_ssl`)的指令有可能不会被列出。

AddModuleInfo 指令

说明	为**server-info**处理器显示的模块增加额外信息
语法	`AddModuleInfo module-name string`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_info
兼容性	仅用于 Apache 1.3 及以上版本

本指令将string的内容作为module-name模块的额外信息以带HTML注解的方式显示。例如：

```
     AddModuleInfo mod_deflate.c 'See <a \
href="http://www.apache.org/docs/2.2/mod/mod_deflate.html">\
       http://www.apache.org/docs/2.2/mod/mod_deflate.html</a>'
```

Apache模块 mod_isapi

说明	仅限于在**Windows**平台上实现**ISAPI**扩展
状态	基本(B)
模块名	isapi_module
源文件	mod_isapi.c
兼容性	仅用于Win32

概述

本模块实现了互联网服务扩展应用程序编程接口(Internet Server extension API)。本模块使得Windows上的Apache能有限地实现互联网服务扩展(比如调用ISAPI的动态连接库)。

ISAPI扩展模块(.dll文件)是由第三方开发的。Apache开发组没有编写这些模块，因此我们也不对它们提供支持。如果在运行ISAPI扩展过程中发生问题，请直接与ISAPI的作者联系。请不要将此类问题贴在**Apache**的邮件列表或错误反馈页面上。

用法

在服务器配置文件中，使用 `AddHandler` 指令将 `isapi-isa` 处理器与ISAPI文件关联起来，并通过文件扩展名来建议对应关系。要将任何一个.dll文件作为ISAPI扩展来处理，需要编辑httpd.conf文件，并加入以下行：

```
AddHandler isapi-isa .dll
```

Apache服务器不允许将服务于请求的模块动态地加载，但可以通过在httpd.conf文件中，加入以下语句使一个模块在Apache启动时预先载入系统，并使其驻留在系统中：

```
ISAPICacheFile c:/WebWork/Scripts/ISAPI/mytest.dll
```

无论是否预载一个ISAPI扩展，所有的ISAPI扩展都使用与CGI脚本相同的许可限制来管理。也就是说，包含ISAPI动态连接库的目录必须要设置" `Options` `ExecCGI` "。

请参阅附加注释和程序员注记以得到关于 `mod_isapi` 所提供的特定的ISAPI支持的细节内容。

附加注释

Apache的ISAPI实现了除部分用来处理异步I/O的微软特定(Microsoft-specific)扩展以外的所有ISAPI 2.0规范。Apache的I/O模型不允许使用ISAPI可能用到的异步读写方式。如果ISAPI试图调用不支持的功能，包括异步I/O，在错误日志中会显示一条错误信息以方便系统的调试。由于这类错误信息可能会大量地产生，指令" `ISAPILogNotSupported Off` "可以使这类错误信息不被记录。

在某些服务器上，比如微软的IIS，ISAPI扩展在载入后将驻留在服务器上，直到内存占用过高，或是指定了不同的配置选项。Apache目前在每次请求时，都会加载和卸载特定的ISAPI扩展，除非指定了 `ISAPICacheFile` 指令。虽然这样看来是效率很低的一种做法，但根据Apache的内存模式使用这种方式是最有效的。许多ISAPI模块与Apache服务器有细微的兼容性问题，卸载这些模块可以保证服务器的稳定运行。

同时请记住Apache支持ISAPI扩展，但它不支持**ISAPI**过滤器。对于ISAPI过滤器的支持可能会在晚些时候加入，但目前没有支持这一功能的计划。

程序员注记

如果你正在开发 Apache 2.0 `mod_isapi` 模块，你必须严格按照以下指令的限制来调用 `ServerSupportFunction` ：

`HSE_REQ_SEND_URL_REDIRECT_RESP`

重定向用户到其它的位置。必须使用完整的、合法的URL(比如： `http://server/location`)。

`HSE_REQ_SEND_URL`

重定向用户到其它的位置。这里不能使用一个完整的URL，你不可以传入协议或服务器名(例如： `/location`)。这类重定向由服务器来处理，不是浏览器。

警告

在最近发布的文档中，微软已经试图放弃这两个 `HSE_REQ_SEND_URL` 函数的差别。但Apache还是将它们视为两个不同的函数加以不同的实现。

`HSE_REQ_SEND_RESPONSE_HEADER`

如果在请求头字符串变量中，请求头的内容后面紧跟一个空行(两个连续的换行)，然后再加上请求体的内容，Apache能接受这一相应的请求体。因为请求头变量是以NULL结束的，这个请求体里不能包含NULL。

`HSE_REQ_DONE_WITH_SESSION`

Apache认为这是一个空操作，因为当ISAPI处理返回时，会话就结束了。

`HSE_REQ_MAP_URL_TO_PATH`

Apache将虚拟名字转换为物理名字。

`HSE_APPEND_LOG_PARAMETER`

这一日志信息可以在以下任一地方捕获：

- 在 `CustomLog` 指令中的 `\"%{isapi-parameter}n\"` 元素里
- 在" `ISAPIAppendLogToQuery` `On` "指令中的" `%q` "日志元素里
- 由" `ISAPIAppendLogToErrors` `On` "指令所产生的错误日志中

第一行的 `%{isapi-parameter}n` 元素总是可用的，并且是推荐的。

`HSE_REQ_IS_KEEP_CONN`

返回Keep-Alive的协商状态。

`HSE_REQ_SEND_RESPONSE_HEADER_EX`

即使 `fKeepConn` 标志被忽略，还是按有证书的方式来处理。

`HSE_REQ_IS_CONNECTED`

如果请求退出则报告错误。

对于所有不支持的 `ServerSupportFunction` 调用，Apache返回 `FALSE` ，同时将 `GetLastError` 的值置为 `ERROR_INVALID_PARAMETER` 。

`ReadClient` 越过初始缓冲区(由 `ISAPIReadAheadBuffer` 指令定义)得到请求的数据包。根据 `ISAPIReadAheadBuffer` (在调用ISAPI处理前缓冲的数据字节数)的设定，较小的请求包当请求被调用时，直接完全地传送到ISAPI扩展。如果请求包很长，ISAPI扩展必须使用 `ReadClient` 得到剩下的请求数据。

支持 `WriteClient` ，但只能使用 `HSE_IO_SYNC` 标志或不带标志(" `0` "值)。任何其它的 `WriteClient` 请求会被拒绝，并且返回 `FALSE` ，同时 `GetLastError` 的值被置为 `ERROR_INVALID_PARAMETER` 。

支持 `GetServerVariable` ，虽然扩展服务变量不存在(定义在其它服务器上)。包括 `ALL_HTTP` 和 `ALL_RAW` ，所有的常规Apache CGI环境变量都可以通过 `GetServerVariable` 得到。

Apache 2.0 `mod_isapi` 支持后来版本的ISAPI规范中的新增功能，比如对异步I/O的有限仿真及 `TransmitFile` 语义。Apache同时也支持ISAPI .dlls 的预载入以提高性能，以上这些在Apache1.3 `mod_isapi` 都没有实现。

ISAPIAppendLogToErrors 指令

说明	把ISAPI扩展的 `HSE_APPEND_LOG_PARAMETER` 请求记录在错误日志中
语法	`ISAPIAppendLogToErrors on|off`
默认值	`ISAPIAppendLogToErrors off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_isapi

把ISAPI扩展的 `HSE_APPEND_LOG_PARAMETER` 请求记录在错误日志中

ISAPIAppendLogToQuery 指令

说明	把ISAPI扩展的 `HSE_APPEND_LOG_PARAMETER` 请求记录在查询域中
语法	`ISAPIAppendLogToQuery on|off`
默认值	`ISAPIAppendLogToQuery on`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_isapi

把ISAPI扩展的 `HSE_APPEND_LOG_PARAMETER` 请求记录在查询域中(追加在 `CustomLog` `%q` 元素后面)。

ISAPICacheFile 指令

说明	启动时载入的ISAPI动态连接库
语法	`ISAPICacheFile file-path [file-path] ...`
作用域	server config, virtual host
状态	基本(B)
模块	mod_isapi

指定一个需在Apache服务启动的时候载入的以空格分隔的文件列表，这些文件驻留在系统中直至服务器关闭。本指令可以为每个需要的ISAPI动态连接库文件所重复。应指定每个文件的路径。如果不是绝对路径，则会基于 `ServerRoot` 来处理相对路径。

ISAPIFakeAsync 指令

说明	为ISAPI回调模拟异步支持
语法	`ISAPIFakeAsync on|off`
默认值	`ISAPIFakeAsync off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_isapi

当设为on时，模拟ISAPI回调的异步支持。

ISAPILogNotSupported 指令

说明	记录ISAPI不支持的功能调用
语法	`ISAPILogNotSupported on|off`
默认值	`ISAPILogNotSupported off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_isapi

在服务错误日志中记录所有ISAPI扩展不支持的请求。本指令可以帮助系统管理员跟踪错误。把这个指令定义为on以后，如果所有的ISAPI模块都工作良好，应该把它设回为Off。

ISAPIReadAheadBuffer 指令

说明	传送到ISAPI扩展的预读缓冲区大小
语法	`ISAPIReadAheadBuffer size`
默认值	`ISAPIReadAheadBuffer 49152`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_isapi

定义初始调用时传送到ISAPI扩展的最大预读缓冲区大小。所有其它的数据必须通过`ReadClient`回调功能得到；部分ISAPI扩展可能不支持`ReadClient`功能。请参考ISAPI扩展本身对相关问题描述。

Apache模块 mod_ldap

说明	为其它LDAP模块提供LDAP连接池和结果缓冲服务
状态	扩展(E)
模块名	ldap_module
源文件	util_ldap.c
兼容性	仅在 Apache 2.0.41 及以后的版本中可用

概述

本模块通过后端连接LDAP服务来改善网站性能。除了标准LDAP库提供的功能外，本模块增加了一个LDAP连接池和一个LDAP共享内存缓冲区。

为了使用本模块的功能，LDAP支持必须编译进APU。这是通过在编译Apache时，在 `configure` 脚本命令行上增加 `--with-ldap` 开关来实现的。

为了支持SSL/TLS，需要APR连接以下一个LDAP SDK ：OpenLDAP SDK(2.x或更新), Novell LDAP SDK, Mozilla LDAP SDK, 本地 Solaris LDAP SDK (基于Mozilla), 本地 Microsoft LDAP SDK, iPlanet (Netscape) SDK 。参见APR网站以获取更多信息。

示例配置

下面的配置是一个使用 `mod_ldap` 模块来提升 `mod_authnz_ldap` 提供的HTTP基本认证性能的例子。

```
        # 开启LDAP连接池及共享内存缓冲。

        # 开启LDAP缓冲状态处理器。需要载入mod_ldap和mod_authnz_ldap模块。

        # 把"yourdomain.example.com"改为你真实的域名。

        LDAPSharedCacheSize 200000

        LDAPCacheEntries 1024

        LDAPCacheTTL 600

        LDAPOpCacheEntries 1024

        LDAPOpCacheTTL 600

        <Location /ldap-status>

   SetHandler ldap-status

          Order deny,allow

          Deny from all

          Allow from yourdomain.example.com

          AuthLDAPEnabled on

          AuthLDAPURL ldap://127.0.0.1/dc=example,dc=com?uid?one

          AuthLDAPAuthoritative on

          require valid-user
        </Location>
```

LDAP连接池

LDAP连接是在请求之间共享的。这就允许LDAP服务器在跳过unbind->connect->rebind这样一个工作周期的情况下，保留连接以减少为下一次请求准备连接的时间。这种性能优化有点象HTTP服务的Keep-Alives功能。

在一个比较繁忙的服务器上，很有可能许多请求同时尝试与同一个LDAP服务进行连接并得到它的服务。如果一个LDAP连接正在使用，Apache会在原来连接的基础上，生成一个新的连接。这将确保连接池不会成为瓶颈。

不需要在Apache配置中手动开启连接池功能。任何使用本模块来访问LDAP服务的模块会自动共享连接池。

LDAP缓冲

为了改善性能， `mod_ldap` 模块使用一种积极的缓冲策略以尽量减少与LDAP服务器的联系。通过缓冲，可以方便地使Apache在提供受mod_authnz_ldap保护的页面时，得到二倍或三倍的吞吐量。同时，LDAP服务器的负载也会明显地减小。

`mod_ldap` 支持两种类型的LDAP缓冲。在search/bind阶段，使用一个*search/bind*缓冲，在compare阶段，使用两个*operation*缓冲。服务器引用的每个LDAP URL都有一组它自己的上述三个缓冲。

Search/Bind缓冲

处理一个查询和绑定操作对LDAP实施来讲，是非常耗时，尤其当目录很大时，这一点更加明显。Search/bind缓冲用来缓冲所有的最终能成功绑定的查询。失败的结果(比如：不成功的查询或查询结果无法成功绑定)不会被缓冲。这样做是因为信任关系失败的连接在所有连接中只占了很小的一个百分比，因此，通过不缓冲这些连接，可以减少缓冲区的大小。

`mod_ldap` 在缓冲区里储存了用户名、得到的DN 、用来绑定的口令、绑定的时间。当一个新的连接用同一个用户名来初始化的时候， `mod_ldap` 将新的连接的口令与保存在缓冲区里的口令进行比较。如果口令匹配，并且那个缓冲项目尚未失效的话， `mod_ldap` 就跳过search/bind阶段。

查询与绑定缓冲由 `LDAPCacheEntries` 和 `LDAPCacheTTL` 指令来控制。

Operation缓冲

在区分与辨别过程中， `mod_ldap` 使用两个操作缓冲区来缓冲比较的操作。第一个缓冲区用来缓冲是否LDAP组成员的测试结果，第二个用来缓冲不同名字间鉴别的比较结果。

这两个缓冲区都是由 `LDAPOpCacheEntries` 和 `LDAPOpCacheTTL` 指令来控制的。

缓冲区的监控

`mod_ldap` 包含了一个完整的处理器，通过它可以使管理员监控缓冲区的性能。这个处理器的名字是 `ldap-status` ，因此可以用下列指令来得到 `mod_ldap` 缓冲区的相关信息：

```
        <Location /server/cache-info>

SetHandler ldap-status
        </Location>
```

通过URL `http://servername/cache-info` ，管理员可以得到 `mod_ldap` 使用的每个缓冲的状态报告。注意，如果Apache不支持共享内存，那么每个 `httpd` 实例都有它自己的缓冲区，因此，每次使用上述URL都可能会得到不同的结果，这取决于具体哪个 `httpd` 实例处理了这个请求。

使用SSL/TSL

通过 `LDAPTrustedGlobalCert` , `LDAPTrustedClientCert` , `LDAPTrustedMode` 指令可以定义与LDAP服务器建立SSL/TSL联接。这些指令指定了使用的CA和可选的客户端证书，以及连接使用的加密类型(none, SSL, TLS/STARTTLS)。

```
# 在636端口建立一个SSL LDAP联接。需要模块mod_ldap和mod_authnz_ldap的支持。

# 将"yourdomain.example.com"修改为您自己的域名。

LDAPTrustedGlobalCert CA_DER /certs/certfile.der

<Location /ldap-status>

SetHandler ldap-status

  Order deny,allow

  Deny from all

  Allow from yourdomain.example.com

  AuthLDAPEnabled on

  AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one

  AuthLDAPAuthoritative on

  require valid-user
</Location>
```

```
# 在389端口建立一个TLS LDAP联接。需要模块mod_ldap和mod_authnz_ldap的支持。

# 将"yourdomain.example.com"修改为您自己的域名。

LDAPTrustedGlobalCert CA_DER /certs/certfile.der

<Location /ldap-status>

SetHandler ldap-status

  Order deny,allow

  Deny from all

  Allow from yourdomain.example.com

  AuthLDAPEnabled on

  LDAPTrustedMode TLS
  AuthLDAPURL ldap://127.0.0.1/dc=example,dc=com?uid?one

  AuthLDAPAuthoritative on

  require valid-user
</Location>
```

SSL/TLS Certificates

The different LDAP SDKs have widely different methods of setting and handling both CA and client side certificates.

If you intend to use SSL or TLS, read this section CAREFULLY so as to understand the differences between configurations on the different LDAP toolkits supported.

Netscape/Mozilla/iPlanet SDK

CA certificates are specified within a file called cert7.db. The SDK will not talk to any LDAP server whose certificate was not signed by a CA specified in this file. If client certificates are required, an optional key3.db file may be specified with an optional password. The secmod file can be specified if required. These files are in the same format as used by the Netscape Communicator or Mozilla web browsers. The easiest way to obtain these files is to grab them from your browser installation.

Client certificates are specified per connection using the LDAPTrustedClientCert directive by referring to the certificate "nickname". An optional password may be specified to unlock the certificate's private key.

The SDK supports SSL only. An attempt to use STARTTLS will cause an error when an attempt is made to contact the LDAP server at runtime.

```
            # Specify a Netscape CA certificate file

            LDAPTrustedGlobalCert CA_CERT7_DB /certs/cert7.db

            # Specify an optional key3.db file for client certificate support

            LDAPTrustedGlobalCert CERT_KEY3_DB /certs/key3.db

            # Specify the secmod file if required

            LDAPTrustedGlobalCert CA_SECMOD /certs/secmod

            <Location /ldap-status>

SetHandler ldap-status

                Order deny,allow

                Deny from all

                Allow from yourdomain.example.com

                AuthLDAPEnabled on

                LDAPTrustedClientCert CERT_NICKNAME <nickname> [password]

                AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one

                AuthLDAPAuthoritative on

                require valid-user
            </Location>
```

Novell SDK

One or more CA certificates must be specified for the Novell SDK to work correctly. These certificates can be specified as binary DER or Base64 (PEM) encoded files.

Note: Client certificates are specified globally rather than per connection, and so must be specified with the LDAPTrustedGlobalCert directive as below. Trying to set client certificates via the LDAPTrustedClientCert directive will cause an error to be logged when an attempt is made to connect to the LDAP server..

The SDK supports both SSL and STARTTLS, set using the LDAPTrustedMode parameter. If an ldaps:// URL is specified, SSL mode is forced, override this directive.

```
# Specify two CA certificate files
LDAPTrustedGlobalCert CA_DER /certs/cacert1.der
LDAPTrustedGlobalCert CA_BASE64 /certs/cacert2.pem
# Specify a client certificate file and key
LDAPTrustedGlobalCert CERT_BASE64 /certs/cert1.pem
LDAPTrustedGlobalCert KEY_BASE64 /certs/key1.pem [password]
# Do not use this directive, as it will throw an error
#LDAPTrustedClientCert CERT_BASE64 /certs/cert1.pem
```

OpenLDAP SDK

One or more CA certificates must be specified for the OpenLDAP SDK to work correctly. These certificates can be specified as binary DER or Base64 (PEM) encoded files.

Client certificates are specified per connection using the LDAPTrustedClientCert directive.

The documentation for the SDK claims to support both SSL and STARTTLS, however STARTTLS does not seem to work on all versions of the SDK. The SSL/TLS mode can be set using the LDAPTrustedMode parameter. If an ldaps:// URL is specified, SSL mode is forced. The OpenLDAP documentation notes that SSL (ldaps://) support has been deprecated to be replaced with TLS, although the SSL functionality still works.

```
             # Specify two CA certificate files
             LDAPTrustedGlobalCert CA_DER /certs/cacert1.der
             LDAPTrustedGlobalCert CA_BASE64 /certs/cacert2.pem
            <Location /ldap-status>
SetHandler ldap-status
                Order deny,allow
                Deny from all
                Allow from yourdomain.example.com
                AuthLDAPEnabled on
                LDAPTrustedClientCert CERT_BASE64 /certs/cert1.pem
                LDAPTrustedClientCert KEY_BASE64 /certs/key1.pem
                AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one
                AuthLDAPAuthoritative on
                require valid-user
            </Location>
```

Solaris SDK

SSL/TLS for the native Solaris LDAP libraries is not yet supported. If required, install and use the OpenLDAP libraries instead.

Microsoft SDK

SSL/TLS certificate configuration for the native Microsoft LDAP libraries is done inside the system registry, and no configuration directives are required.

Both SSL and TLS are supported by using the ldaps:// URL format, or by using the LDAPTrustedMode directive accordingly.

Note: The status of support for client certificates is not yet known for this toolkit.

LDAPCacheEntries 指令

说明	主LDAP缓冲的最大条目数
语法	`LDAPCacheEntries number`
默认值	`LDAPCacheEntries 1024`
作用域	server config
状态	扩展(E)
模块	mod_ldap

指定主LDAP缓冲的最大条目数。这个缓冲区包含了成功的search/bind对。把它设为0可以关闭search/bind缓冲。默认值是1024 。

LDAPCacheTTL 指令

说明	search/bind缓冲项目有效时限
语法	`LDAPCacheTTL seconds`
默认值	`LDAPCacheTTL 600`
作用域	server config
状态	扩展(E)
模块	mod_ldap

指定search/bind缓冲项目有效的时间，以秒为单位。默认为600秒(10分钟)。

LDAPConnectionTimeout 指令

说明	指定套接字连接超时秒数
语法	`LDAPConnectionTimeout seconds`
作用域	server config
状态	扩展(E)
模块	mod_ldap

Specifies the timeout value (in seconds) in which the module will attempt to connect to the LDAP server. If a connection is not successful with the timeout period, either an error will be returned or the module will attempt to connect to a secondary LDAP server if one is specified. The default is 10 seconds.

LDAPOpCacheEntries 指令

说明	LDAP compare缓冲区的大小
语法	`LDAPOpCacheEntries number`
默认值	`LDAPOpCacheEntries 1024`
作用域	server config
状态	扩展(E)
模块	mod_ldap

指定 `mod_ldap` 使用的LDAP compare缓冲区大小。默认值是1024条。把它设为0可以关闭操作缓冲。

LDAPOpCacheTTL 指令

说明	操作缓冲有效时限
语法	`LDAPOpCacheTTL seconds`
默认值	`LDAPOpCacheTTL 600`
作用域	server config
状态	扩展(E)
模块	mod_ldap

指定操作缓冲项目的有效时长，以秒为单位。默认为600秒。

LDAPSharedCacheFile 指令

说明	设置共享内存缓冲区文件
语法	`LDAPSharedCacheFile directory-path/filename`
作用域	server config
状态	扩展(E)
模块	mod_ldap

设置共享内存缓冲区文件。若未设置，将使用匿名共享内存(若平台支持)。

LDAPSharedCacheSize 指令

说明	共享内存缓冲区的字节大小
语法	`LDAPSharedCacheSize bytes`
默认值	`LDAPSharedCacheSize 102400`
作用域	server config
状态	扩展(E)
模块	mod_ldap

指定共享内存缓冲区的大小，以Byte为单位。默认为100KB。

LDAPTrustedClientCert 指令

说明	**Sets the file containing or nickname referring to a per connection client certificate. Not all LDAP toolkits support per connection client certificates.**
语法	`LDAPTrustedClientCert type directory-path/filename/nickname [password]`
作用域	server config, virtual host, directory, .htaccess
状态	扩展(E)
模块	mod_ldap

It specifies the directory path, file name or nickname of a per connection client certificate used when establishing an SSL or TLS connection to an LDAP server. Different locations or directories may have their own independant client certificate settings. Some LDAP toolkits (notably Novell) do not support per connection client certificates, and will throw an error on LDAP server connection if you try to use this directive (Use the LDAPTrustedGlobalCert directive instead for Novell client certificates - See the SSL/TLS certificate guide above for details). The type specifies the kind of certificate parameter being set, depending on the LDAP toolkit being used. Supported types are:

- CERT_DER - binary DER encoded client certificate
- CERT_BASE64 - PEM encoded client certificate
- CERT_NICKNAME - Client certificate "nickname" (Netscape SDK)
- KEY_DER - binary DER encoded private key
- KEY_BASE64 - PEM encoded private key

LDAPTrustedGlobalCert 指令

说明	Sets the file or database containing global trusted Certificate Authority or global client certificates
语法	`LDAPTrustedGlobalCert type directory-path/filename [password]`
作用域	server config
状态	扩展(E)
模块	mod_ldap

It specifies the directory path and file name of the trusted CA certificates and/or system wide client certificates `mod_ldap` should use when establishing an SSL or TLS connection to an LDAP server. Note that all certificate information specified using this directive is applied globally to the entire server installation. Some LDAP toolkits (notably Novell) require all client certificates to be set globally using this directive. Most other toolkits require clients certificates to be set per Directory or per Location using LDAPTrustedClientCert. If you get this wrong, an error may be logged when an attempt is made to contact the LDAP server, or the connection may silently fail (See the SSL/TLS certificate guide above for details). The type specifies the kind of certificate parameter being set, depending on the LDAP toolkit being used. Supported types are:

- CA_DER - binary DER encoded CA certificate
- CA_BASE64 - PEM encoded CA certificate
- CA_CERT7_DB - Netscape cert7.db CA certificate database file
- CA_SECMOD - Netscape secmod database file
- CERT_DER - binary DER encoded client certificate
- CERT_BASE64 - PEM encoded client certificate
- CERT_KEY3_DB - Netscape key3.db client certificate database file
- CERT_NICKNAME - Client certificate "nickname" (Netscape SDK)
- CERT_PFX - PKCS#12 encoded client certificate (Novell SDK)
- KEY_DER - binary DER encoded private key
- KEY_BASE64 - PEM encoded private key
- KEY_PFX - PKCS#12 encoded private key (Novell SDK)

LDAPTrustedMode 指令

说明	**Specifies the SSL/TLS mode to be used when connecting to an LDAP server.**
语法	`LDAPTrustedMode type`
作用域	server config, virtual host, directory, .htaccess
状态	扩展(E)
模块	mod_ldap

The following modes are supported:

- NONE - no encryption
- SSL - ldaps:// encryption on default port 636
- TLS - STARTTLS encryption on default port 389

Not all LDAP toolkits support all the above modes. An error message will be logged at runtime if a mode is not supported, and the connection to the LDAP server will fail.

If an ldaps:// URL is specified, the mode becomes SSL and the setting of LDAPTrustedMode is ignored.

LDAPVerifyServerCert 指令

说明	**Force server certificate verification**
语法	`LDAPVerifyServerCert On|Off`
默认值	`LDAPVerifyServerCert On`
作用域	server config
状态	扩展(E)
模块	mod_ldap

Specifies whether to force the verification of a server certificate when establishing an SSL connection to the LDAP server.

Apache模块 mod_log_config

说明	允许记录日志和定制日志文件格式
状态	基本(B)
模块名	log_config_module
源文件	mod_log_config.c

概述

本模块提供了灵活的方法将客户请求记录到日志。日志可以用自定义的格式直接写入文件，或者传送到一个外部程序继续处理。条件日志功能可以实现根据请求的特征来决定一个日志信息是否被包含在最终的日志记录里面。

本模块提供了三个指令： `TransferLog` 指令用来指定日志文件， `LogFormat` 指令用来定义日志格式， `CustomLog` 指令可以同时完成指定日志文件和定义日志格式。 `TransferLog` 和 `CustomLog` 指令在每个服务器上都可以被多次使用，以便将同一个请求记录到多个文件中。

定制日志文件格式

`LogFormat` 和 `CustomLog` 指令的格式化参数是一个字符串。这个字符串会在每次请求发生的时候，被记录到日志中去。它可以包含将被原样写入日志的文本字符串以及C风格的控制字符"\n"和"\t"以实现换行与制表。文本中的引号和反斜杠应通过"\"来转义。

请求本身的情况将通过在格式字符串中放置各种" `%` "转义符的方法来记录，它们在写入日志文件时，根据下表的定义进行转换：

格式字符串	描述
`%%`	百分号(*Apache2.0.44*或更高的版本)
`%a`	远端IP地址
`%A`	本机IP地址
`%B`	除HTTP头以外传送的字节数
`%b`	以CLF格式显示的除HTTP头以外传送的字节数，也就是当没有字节传送时显示' `-` '而不是0。
`%{Foobar}C`	在请求中传送给服务端的cookieFoobar的内容。
`%D`	服务器处理本请求所用时间，以微为单位。

`%{FOOBAR}e`	环境变量FOOBAR的值
`%f`	文件名
`%h`	远端主机
`%H`	请求使用的协议
`%{Foobar}i`	发送到服务器的请求头 `Foobar:` 的内容。
`%l`	远端登录名(由identd而来，如果支持的话)，除非 `IdentityCheck` 设为" `On` "，否则将得到一个"-"。
`%m`	请求的方法
`%{Foobar}n`	来自另一个模块的注解 `Foobar` 的内容。
`%{Foobar}o`	应答头 `Foobar:` 的内容。
`%p`	服务器服务于该请求的标准端口。
`%P`	为本请求提供服务的子进程的PID。
`%{format}P`	服务于该请求的PID或TID(线程ID)， `format` 的取值范围为： `pid` 和 `tid` (*2.0.46*及以后版本)以及 `hextid` (需要APR1.2.0及以上版本)
`%q`	查询字符串(若存在则由一个" `?` "引导，否则返回空串)
`%r`	请求的第一行
`%s`	状态。对于内部重定向的请求，这个状态指的是原始请求的状态，---- `%>s` 则指的是最后请求的状态。
`%t`	时间，用普通日志时间格式(标准英语格式)
`%{format}t`	时间，用 `strftime(3)` 指定的格式表示的时间。(默认情况下按本地化格式)
`%T`	处理完请求所花时间，以秒为单位。
`%u`	远程用户名(根据验证信息而来；如果返回status(`%s`)为401，可能是假的)
`%U`	请求的URL路径，不包含查询字符串。
`%v`	对该请求提供服务的标准 `ServerName` 。
`%V`	根据 `UseCanonicalName` 指令设定的服务器名称。
`%X`	请求完成时的连接状态： `X` = 连接在应答完成前中断。 `+` = 应答传送完后继续保持连接。 `-` = 应答传送完后关闭连接。 (在1.3以后的版本中，这个指令是 `%c` ，但这样就和过去的SSL语法： `%{var}c` 冲突了)
`%I`	接收的字节数，包括请求头的数据，并且不能为零。要使用这个指令你必须启用 `mod_logio` 模块。
`%O`	发送的字节数，包括请求头的数据，并且不能为零。要使用这个指令你必须启用 `mod_logio` 模块。

修饰符

可以紧跟在"%"后面加上一个逗号分隔的状态码列表来限制记录的条目。例如，" `%400,501{User-agent}i` "只记录状态码400和501发生时的 `User-agent` 头内容；不满足条件时用" `-` "代替。状态码前还可以加上" `!` "前缀表示否定，" `%!200,304,302{Referer}i` "记录所有_不同于_200,304,302的状态码发生时的 `Referer` 头内容。

"<"和">"修饰符可以用来指定对于已被内部重定向的请求是选择原始的请求还是选择最终的请求。默认情况下， `%s, %U, %T, %D, %r` 使用原始请求，而所有其他格式串则选择最终请求。例如， `%>s` 可以用于记录请求的最终状态，而 `%<u` 则记录一个已经被内部重定向到非认证资源的请求的原始认证用户。

一些说明

出于安全考虑，从2.0.46版本开始， `%r` , `%i` , `%o` 中的特殊字符，除了双引号(")和反斜线()分别用 `\"` 和 `\\` 进行转义、空白字符用C风格(`\n` , `\t` 等)进行转义以外，非打印字符和其它特殊字符使用 `\xhh` 格式进行转义(hh是该字符的16进制编码)。在2.0.46以前的版本中，这些内容会被完整的按原样记录。这种做法将导致客户端可以在日志中插入控制字符，所以你在处理这些日志文件的时候要特别小心。

在2.0版本中(不同于1.3)， `%b` 和 `%B` 格式字符串并不表示发送到客户端的字节数，而只是简单的表示HTTP应答字节数(在连接中断或使用SSL时与前者有所不同)。 `mod_logio` 提供的 `%O` 格式字符串将会记录发送的实际字节数。

示例

一些常见的格式串：

通用日志格式(CLF)

```
"%h %l %u %t \"%r\" %&gt;s %b"
```

带虚拟主机的通用日志格式

```
"%v %h %l %u %t \"%r\" %&gt;s %b"
```

NCSA扩展/组合日志格式

```
"%h %l %u %t \"%r\" %&gt;s %b \"%{Referer}i\" \"%{User-agent}i\""
```

Referer日志格式

```
"%{Referer}i -&gt; %U"
```

Agent(Browser)日志格式

```
"%{User-agent}i"
```

安全考虑

如果放置日志文件的目录对除启动Apache服务以外的其他用户可写，可能会对系统的安全性造成威胁，具体的讨论请参见安全方面的提示。

BufferedLogs 指令

说明	在将日志写入磁盘前先在内存中进行缓冲
语法	`BufferedLogs On|Off`
默认值	`BufferedLogs Off`
作用域	server config
状态	基本(B)
模块	mod_log_config
兼容性	仅在 Apache 2.0.41 及以后的版本中可用

`BufferedLogs` 指令使得 `mod_log_config` 先在内存中缓冲一些日志内容，然后一次性写入磁盘，而不是立即写入。在一些系统上这样做可以提高磁盘性能。这个设置仅能够针对全局进行设置，不能单独针对虚拟主机进行设置。

这是一个试验性的指令，请在使用中多加小心。

CookieLog 指令

说明	设定针对**cookies**的日志文件名
语法	`CookieLog filename`
作用域	server config, virtual host
状态	基本(B)
模块	mod_log_config
兼容性	反对使用该指令

`CookieLog` 指令使用cookies作为日志文件名。文件是相对于 `ServerRoot` 目录的。包含本指令仅仅是为了保持与 `mod_cookies` 模块的兼容，并且反对使用。

CustomLog 指令

说明	设定日志的文件名和格式
语法	`CustomLog file|pipe format|nickname [env=[!]environment-variable]`
作用域	server config, virtual host
状态	基本(B)
模块	mod_log_config

`CustomLog` 指令用来对服务器的请求进行日志记录。可以指定日志的格式，也可以使用环境变量根据请求的特征来自由地组织日志。

第一个参数指定了日志记录的位置，可以使用以下两种方式来设定：

file

相对于 `ServerRoot` 的日志文件名。

pipe

管道符" `|` "后面紧跟着一个把日志输出当作标准输入的处理程序路径。

安全

如果这里用到了程序，那么这个程序是以启动 `httpd` 的用户来执行的。因此如果启动httpd的用户是root ，那这个程序也将以root身份来运行；你需要确认这个程序是安全的。

注意

当在非UNIX平台上输入文件路径的时候，要特别注意即使平台本身是使用反斜杠()来分隔路径的，在这里也只能使用正斜杠(/)。通常在配置文件里只用正斜杠(/)来分隔路径总是不会错的。

第二个参数指定了写入日志文件的内容。它既可以是由前面的 `LogFormat` 指令定义的nickname ，也可以是直接按日志格式一节所描述的规则定义的format字符串。

例如：以下两组指令的结果是完全一样的：

```
# 使用nickname
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
# 明确使用格式格式字符串
CustomLog logs/access_log "%h %l %u %t \"%r\" %>s %b"
```

第三个参数是可选的，它根据服务器上特定的环境变量是否被设置来决定是否对某一特定的请求进行日志记录。如果这个特定的环境变量被设置(或者在" `env=!name` "的情况下未被设置)，那么这个请求将被记录。

可以使用 `mod_setenvif` 和/或 `mod_rewrite` 模块来为每个请求设置环境变量。例如：如果你想在服务器上将所有对GIF图片的请求记录在不同于主日志文件的另一个日志文件中，你可以使用下面的指令：

```
SetEnvIf Request_URI \.gif$ gif-image
CustomLog gif-requests.log common env=gif-image
CustomLog nongif-requests.log common env=!gif-image
```

或者为了复制旧有的RefererIgnore指令的行为，你可以使用下面的指令：

```
SetEnvIf Referer example\.com localreferer
CustomLog referer.log referer env=!localreferer
```

LogFormat 指令

说明	定义访问日志的记录格式
语法	LogFormat format|nickname [nickname]
默认值	LogFormat "%h %l %u %t \"%r\" %>s %b"
作用域	server config, virtual host
状态	基本(B)
模块	mod_log_config

本指令定义访问日志的记录格式。

`LogFormat` 指令可以使用两种定义格式中的一种。在第一种格式中，指令只带一个参数，以定义后续的 `TransferLog` 指令定义的日志格式。这个唯一的参数可以按上述自定义日志格式小节所描述的format来定义。另外它也可以通过下述的方法使用nickname来引用某个之前的 `LogFormat` 定义的日志格式。

第二种定义 `LogFormat` 指令的格式中，将一个直接的format和一个nickname联系起来。这样在后续的 `LogFormat` 或 `CustomLog` 指令中，就不用一再重复整个冗长的格式串。定义别名的 `LogFormat` 指令仅仅用来定义一个nickname， 而不做其它任何事情：也就是说， 它只是定义了这个别名，它既没有实际应用这个别名，也不是把它设为默认的格式。因此，它不会影响后续的 `TransferLog` 指令。另外， `LogFormat` 不能用一个别名来定义另一个别名。注意， 别名不能包含百分号(`%`)。

示例

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b" vhost_common
```

TransferLog 指令

说明	指定日志文件的位置
语法	`TransferLog file|pipe`
作用域	server config, virtual host
状态	基本(B)
模块	mod_log_config

本指令除不允许直接定义日志格式或根据条件进行日志记录外，与 `CustomLog` 指令有完全相同的参数和功能。实际应用中，日志的格式是由最近的非别名定义的 `LogFormat` 指令指定。如果没有定义任何日志格式，则使用通用日志格式。

示例

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""

TransferLog logs/access_log
```

Apache模块 mod_log_forensic

说明	实现"对比日志"，即在请求被处理之前和处理完成之后进行两次记录
状态	扩展(E)
模块名	log_forensic_module
源文件	mod_log_forensic.c
兼容性	`mod_unique_id` is no longer required since version 2.1

概述

This module provides for forensic logging of client requests. Logging is done before and after processing a request, so the forensic log contains two log lines for each request. The forensic logger is very strict, which means:

- The format is fixed. You cannot modify the logging format at runtime.
- If it cannot write its data, the child process exits immediately and may dump core (depending on your `CoreDumpDirectory` configuration).

`check_forensic` script, which can be found in the distribution's support directory, may be helpful in evaluating the forensic log output.

Forensic Log Format

Each request is logged two times. The first time is *before* it's processed further (that is, after receiving the headers). The second log entry is written *after* the request processing at the same time where normal logging occurs.

In order to identify each request, a unique request ID is assigned. This forensic ID can be cross logged in the normal transfer log using the `%{forensic-id}n` format string. If you're using `mod_unique_id`, its generated ID will be used.

The first line logs the forensic ID, the request line and all received headers, separated by pipe characters (`|`). A sample line looks like the following (all on one line):

```
+yQtJf8CoAB4AAFNXBIEAAAAA|GET /manual/de/images/down.gif
HTTP/1.1|Host:localhost%3a8080|User-Agent:Mozilla/5.0 (X11;
U; Linux i686; en-US; rv%3a1.6) Gecko/20040216
Firefox/0.8|Accept:image/png, <var class="calibre40">etc...</var>
```

The plus character at the beginning indicates that this is the first log line of this request. The second line just contains a minus character and the ID again:

```
-yQtJf8CoAB4AAFNXBIEAAAAA
```

`check_forensic` script takes as its argument the name of the logfile. It looks for those `+` / `-` ID pairs and complains if a request was not completed.

Security Considerations

See the security tips document for details on why your security could be compromised if the directory where logfiles are stored is writable by anyone other than the user that starts the server.

ForensicLog 指令

说明	Sets filename of the forensic log
语法	`ForensicLog filename|pipe`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_log_forensic

`ForensicLog` directive is used to log requests to the server for forensic analysis. Each log entry is assigned a unique ID which can be associated with the request using the normal `CustomLog` directive. `mod_log_forensic` creates a token called `forensic-id`, which can be added to the transfer log using the `%{forensic-id}n` format string.

The argument, which specifies the location to which the logs will be written, can take one of the following two types of values:

filename

A filename, relative to the `ServerRoot`.

pipe

The pipe character "`|`", followed by the path to a program to receive the log information on its standard input. The program name can be specified relative to the `ServerRoot` directive.

安全

If a program is used, then it will be run as the user who started `httpd`. This will be root if the server was started by root; be sure that the program is secure or switches to a less privileged user.

注意

When entering a file path on non-Unix platforms, care should be taken to make sure that only forward slashed are used even though the platform may allow the use of back slashes. In general it is a good idea to always use forward slashes throughout the configuration files.

Apache模块 mod_logio

说明	对每个请求的输入/输出字节数以及**HTTP**头进行日志记录
状态	扩展(E)
模块名	logio_module
源文件	mod_logio.c

概述

本模块可以对每个请求的输入/输出字节数进行日志记录。这个字节数反映网络上实际传输的字节数，它包括了请求头与响应头正文的字节数之和。输入计数在SSL/TLS之前进行，输出计数在SSL/TLS之后进行，因此计数能正确反映加密所造成的影响。

本模块需要 `mod_log_config` 模块的支持。

自定义日志格式

本模块加入了两个新的日志指令。请求特征由在格式字符串里加"`%`"指令完成。在日志文件中，它的值按以下方式被记录：

格式字符串	说明
`%I`	接收的字节数，包含头与正文，不能为零。
`%O`	发送的字节数，包含头与正文，不能为零。

通常，本功能按下述方式使用：

组合I/O日志格式：

```
"%h %l %u %t \"%r\" %&gt;s %b \"%{Referer}i\" \"%{User-agent}i\" %I %O"
```

Apache模块 mod_mem_cache

说明	基于内存的缓冲管理器
状态	扩展(E)
模块名	mem_cache_module
源文件	mod_mem_cache.c

概述

本模块需要 `mod_cache` 提供的服务。它作为 `mod_cache` 的辅助模块工作，以提供基于内存的存储管理。 `mod_mem_cache` 模块可以按两种方法来配置：缓存打开的文件描述符，或缓存堆中的对象。 `mod_mem_cache` 大多数情况下用于缓存本地生成的内容，或是在 `mod_proxy` 配置成 `ProxyPass` 方式(也就是<dfn class="calibre27">反向代理</dfn>)时后端服务器产生的内容。

被缓存的内容通过基于URI的键被保存到缓冲区或从缓冲区取出。带访问保护的内容不会被缓存。

MCacheMaxObjectCount 指令

说明	最大缓存对象数
语法	`MCacheMaxObjectCount value`
默认值	`MCacheMaxObjectCount 1009`
作用域	server config
状态	扩展(E)
模块	mod_mem_cache

`MCacheMaxObjectCount` 指令设定最大缓存对象数。这个值用来生成开放哈希表。如果一个新的对象要被插入缓存，而此时缓存的最大缓存对象数已达到极限，原来缓存的某个对象会被移走以允许新对象插入缓存。具体哪个对象被移走，是通过 `MCacheRemovalAlgorithm` 指令指定的算法来确定的。

示例

```
MCacheMaxObjectCount 13001
```

MCacheMaxObjectSize 指令

说明	缓存允许的最大文档大小(字节)
语法	`MCacheMaxObjectSize bytes`
默认值	`MCacheMaxObjectSize 10000`
作用域	server config
状态	扩展(E)
模块	mod_mem_cache

`MCacheMaxObjectSize` 指令设定允许缓存的最大文档大小(以Byte为单位)。

示例

```
MCacheMaxObjectSize 6400000
```

注意

`MCacheMaxObjectSize` 的值必须大于 `MCacheMinObjectSize` 的值。

MCacheMaxStreamingBuffer 指令

说明	内存中允许缓冲的最大流式响应字节长度
语法	`MCacheMaxStreamingBuffer size_in_bytes`
默认值	`MCacheMaxStreamingBuffer 100000与MCacheMaxObjectSize中的小者`
作用域	server config
状态	扩展(E)
模块	mod_mem_cache

`MCacheMaxStreamingBuffer` 内存中允许缓冲的最大流式响应字节长度，以决定这个流式响应是否太长而不能被缓存。流式响应是指整个响应内容无法完整地得到，并且 `Content-Length` 也未知的响应。流式响应的来源包括代理的响应内容和CGI脚本的输出。默认情况下，除非响应头包含 `Content-Length` 信息，否则流式响应不会被缓存。这样做是为了避免使用大量内存缓存那些最终因为太长而不能缓存的响应。而 `MCacheMaxStreamingBuffer` 指令允许缓存不含 `Content-Length` 的流式响应，直到达到其指定的值。如果达到了最大流式响应可缓冲长度，已经被缓存的内容将会被释放，缓存也不再继续。

注意：

对 `MCacheMaxStreamingBuffer` 使用一个非零值不会对响应传送到客户产生延迟。当 `mod_mem_cache` 模块将流式化的内容复制到缓冲区的同时会将数据块传送到下一个输入点以传送给客户端。

```
# 指定最大流式响应可缓冲长度为 64KB
MCacheMaxStreamingBuffer 65536
```

MCacheMinObjectSize 指令

说明	允许缓存的最小文档大小(字节)
语法	`MCacheMinObjectSize bytes`
默认值	`MCacheMinObjectSize 0`
作用域	server config
状态	扩展(E)
模块	mod_mem_cache

`MCacheMinObjectSize` 指令设定允许缓存的最小文档大小。

示例

```
MCacheMinObjectSize 10000
```

MCacheRemovalAlgorithm 指令

说明	定义在需要时哪个文档被移出缓存的算法
语法	`MCacheRemovalAlgorithm LRU|GDSF`
默认值	`MCacheRemovalAlgorithm GDSF`
作用域	server config
状态	扩展(E)
模块	mod_mem_cache

`MCacheRemovalAlgorithm` 指令定义在需要时哪个文档被移出缓存。可以有两种选择：

`LRU` (最近最少使用)

`LRU` 指定最长时间没有用到的对象将在必要的时候移出缓存。

`GDSF` (GreadyDual-Size)

`GDSF` 基于缓存命中率和文档大小计算优先级。在必要时，优先级最低的文档被移出缓存。

示例

```
MCacheRemovalAlgorithm GDSF
MCacheRemovalAlgorithm LRU
```

MCacheSize 指令

说明	缓存允许使用的最大内存量，以**KB**为单位
语法	`MCacheSize KBytes`
默认值	`MCacheSize 100`
作用域	server config
状态	扩展(E)
模块	mod_mem_cache

`MCacheSize` 指令设定缓存允许使用的最大内存量，以KB(1024-byte)为单位。如果一个新的比缓存剩余内存量大的对象要插入缓存，那么原来缓存内的对象会被移走，直到这个新的对象能被插入缓冲区。具体哪个对象被移走，是通过 `MCacheRemovalAlgorithm` 指令指定的算法来确定的。

示例

```
MCacheSize 700000
```

注意

`MCacheSize` 的值必须比 `MCacheMaxObjectSize` 的值大。

Apache模块 mod_mime

说明	根据文件扩展名决定应答的行为(处理器/过滤器)和内容(MIME类型/语言/字符集/编码)
状态	基本(B)
模块名	mime_module
源文件	mod_mime.c

概述

本模块通过文件的扩展名将不同的"元信息"与文件关联起来。元信息在文档的文件名与文档的MIME类型、语言、字符集、编码方式之间建立关联。最终元信息会传送到服务器并参与内容协商，这样最终在考虑用户指定参数的基础上，在几个可能的文件里选择一个提供服务。关于内容协商的更多信息，请参阅 `mod_negotiation` 模块。

`AddCharset` ， `AddEncoding` ， `AddLanguage` ， `AddType` 指令都可以用于在文件的扩展名与文件的元信息之间建立映射关系。它们分别指明了文档的字符集、编码方式、语言、MIME类型(内容类型)。指令 `TypesConfig` 用来指定一个文件，它也包含了扩展名到MIME类型的映射关系。

另外， `mod_mime` 还可以定义处理器和过滤器来生成或处理信息。指令 `AddHandler` ， `AddOutputFilter` ， `AddInputFilter` 控制了提供文档的模块或脚本的运作方式。 `MultiviewsMatch` 指令设定 `mod_negotiation` 模块在尝试Multiview匹配时，如何处理文件扩展名。

当 `mod_mime` 模块在元信息与文件的扩展名之间建立映射以后， `core` 提供了一组指令用来建立某个给定范围内(也就是 `<Location>` ， `<Directory>` ， `<Files>`)所有相关文件与特定的元信息之间的关联。这些指令包括 `ForceType` ， `SetHandler` ， `SetInputFilter` ， `SetOutputFilter` 。 `core` 的指令会覆盖任何在 `mod_mime` 模块中定义的文件扩展名映射。

注意，改变一个文件的元信息，不会改变 `Last-Modified` 头的值。因此，以前被缓存的副本可能还会被用户或代理服务器使用。如果你改变了元信息(语言、内容类型、字符集、编码方式)，你需要"触及"所有相关文件(更新他们的最后修改时间)，以保证所有的访问者都收到正确的内容标题。

带多扩展名的文件

文件可以有多个扩展名，这些扩展名的顺序一般情况下是无关紧要的。例如：如果文件 `welcome.html.fr` 被映射为内容类型是 `text/html` 、语言是法语的话，文件 `welcome.fr.html` 将被映射为完全相同的内容。如果一个以上的扩展名映射到同种类型的元信息上，那么将使用最右边的那个。比如： `.gif` 的MIME类型是 `image/gif` 、 `.html` 的MIME类型是 `text/html` ，那么 `welcome.gif.html` 的MIME类型将是 `text/html` 。

语言和内容编码会按照积累的方式处理，因为一个文件可以被指定为多种语言或编码。因此， `welcome.html.en.de` 文件将会按照 `Content-Language: en, de` 和 `Content-Type: text/html` 发送。

在处理带多个扩展名的文件并且这些扩展名同时关联MIME类型和处理器时，要特别小心。这种情况下通常是由与处理器相关的模块来处理得到结果。比如，文件扩展名 `.imap` (通过 `mod_imagemap` 模块)与 `imap-file` 处理器相关联，同时， `.html` 文件扩展名的MIME类型是 `text/html` ，那么文件 `world.imap.html` 将同时与 `imap-file` 处理器和 `text/html` MIME类型`模块处理的图像映射文件。

内容编码

一个具有特定MIME类型的文件能够用一种特定的方法进行额外的编码，以简化它在互联网上的传输。这通常指的是压缩，比如 `gzip` ；也可以是加密，例如 `pgp` ；还可以是像UUencoding那样的编码，UUencoding是用来在ASCII(文本)格式的文件里，传输二进制文件的编码方式。

HTTP/1.1 RFC第14.11节是这样解释的：

> 实体头的"Content-Encoding"域是媒体类型的修饰符。如果存在，它的值指明了对实体本身进行额外编码的方式，以及为了得到"Content-Type"头所参照的媒体类型而必须采用的解码机制。"Content-Encoding"主要用来允许一个文件在不破坏它底层媒体类型的基础上，进行压缩。

通过使用一种以上的文件扩展名(参见上面关于带多扩展名的文件一节)，你可以指定文件是一种特定的类型，还可以同时指定它特定的编码方式。

例如，你有一个文件，它是Microsoft Word文档，同时为了减小它的大小，它还被压缩了。如果 `.doc` 扩展名表示Microsoft Word文件类型，而 `.zip` 扩展名表示pkzip文件编码方式，那么文件 `Resume.doc.zip` 就会被认出是一个用pkzip压缩过的Word文档。

Apache把一个 `Content-encoding` 头和请求的资源一起发送，以便告诉浏览器资源编码的方式。

```
Content-encoding: pkzip
```

字符集和语言

除了文件类型和文件编码方式外，还有一个重要的信息是文件的语言以及文件显示时的字符集。例如一个文档可能是用越南语或古斯拉夫语写的，并且也应该显示成这种语言。这样的信息也要在HTTP头里进行传输。

字符集、语言、编码方式、内容类型等信息都是用在内容协商(参阅 `mod_negotiation` 模块)处理过程中的。它们决定了当许多包含了不同的字符集、语言、编码方式、内容类型的文档都存在时，具体将哪个文档返回给客户端。所有由 `AddCharset` , `AddEncoding` , `AddLanguage` , `AddType` 指令定义的文件扩展名关联(还有在 `MimeMagicFile` 指令中列出的文件扩展名)都参与了这个选择过程。使用 `AddHandler` , `AddInputFilter` , `AddOutputFilter` 指令建立的关联，可以用 `MultiviewsMatch` 指令来决定参与或不参与匹配。

字符集

为了传递更深层次的信息，Apache在传送一个 `Content-Language` 头以指定文档语言的基础上，还在随后的 `Content-Type` 头中指明了具体的字符集，以便更精确地描述这一信息。

```
        Content-Language: en, fr

    Content-Type: text/plain; charset=ISO-8859-1
```

语言的标识是这个语言名字的二字母缩写。 `charset` 是使用的字符集的精确名字。

AddCharset 指令

说明	在给定的文件扩展名与特定的字符集之间建立映射
语法	`AddCharset charset extension [extension] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime

`AddCharset` 指令在特定的文件扩展名与特定的字符集之间建立映射。charset是以extension为扩展名的文件的MIME字符集参数。这个映射关系会强制添加在所有现存的映射关系上，并覆盖所有现存的extension扩展名映射。

示例

```
AddLanguage ja .ja
AddCharset EUC-JP .euc
AddCharset ISO-2022-JP .jis
AddCharset SHIFT_JIS .sjis
```

有了以上定义以后，文档 `xxxx.ja.jis` 会被当成是使用字符集 `ISO-2022-JP` 的日文文档(文档 `xxxx.jis.ja` 也一样)。 `AddCharset` 指令除了用于通知客户端文档的字符集编码方式以便正确地翻译和显示以外，还用于内容协商(根据用户的优先选择信息，从几个文档中选择一个返回给用户)。

extension参数是大小写无关的，并且可以带或不带前导点。

参见

- `mod_negotiation`
- `AddDefaultCharset`

AddEncoding 指令

说明	在文件扩展名与特定的编码方式之间建立映射关系
语法	`AddEncoding MIME-enc extension [extension] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime

`AddEncoding` 指令在文件扩展名与特定的编码方式之间建立映射关系。指令定义以extension为扩展名的文件是由MIME-enc方式编码的。这个映射关系会添加在所有有效的映射关系上，并覆盖所有相同的extension扩展名映射。

示例

```
AddEncoding x-gzip .gz
AddEncoding x-compress .Z
```

有了上述定义后，包含 `.gz` 扩展名的文件被认为是用 `x-gzip` 方式编码的，而带 `.Z` 扩展名的文件则被认为是用 `x-compress` 方式编码的。

老的客户端期望 `x-gzip` 和 `x-compress` ，然而，按标准来说，它们分别等同于 `gzip` 和 `compress` 。Apache在进行编码方式映射时，会忽略" `x-` "前缀。当响应需要包含编码方式时，Apache会使用客户端请求的格式(例如： `x-foo` 或 `foo`)来应答。如果客户端没有指明特定的格式，Apache会使用 `AddEncoding` 指令给定的编码方式。为了简化这一问题，你应该为这两个特定的编码方式始终使用 `x-gzip` 和 `x-compress` 。对于象 `deflate` 这样比较新的编码方式，指定时不要带" `x-` "前缀。

extension参数是大小无关的，并且可以带或不带前导点。

AddHandler 指令

说明	在文件扩展名与特定的处理器之间建立映射
语法	`AddHandler handler-name extension [extension] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime

指定带extension扩展名的文件应被handler-name处理器来处理。这个映射关系会添加在所有有效的映射关系上，并覆盖所有相同的extension扩展名映射。例如，为了把扩展名为 `.cgi` 的文件作为CGI脚本来处理，你应该定义：

```
AddHandler cgi-script .cgi
```

一旦将上述定义放在你的http.conf文件中，所有包含 `.cgi` 扩展名的文件，都会被当成是CGI程序。

extension参数是大小无关的，并且可以带或不带前导点。

参见

- `SetHandler`

AddInputFilter 指令

说明	在文件扩展名与特定的输入过滤器之间建立映射
语法	`AddInputFilter filter[;filter...] extension [extension] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime
兼容性	仅在 Apache 2.0.26 及以后的版本中可用

`AddInputFilter` 指令在文件扩展名extension与对服务器收到的客户请求进行处理与转发的输入过滤器之间建立映射。这是除包括 `SetInputFilter` 指令在内的所有过滤器定义指令之外的定义。这个映射会与所有有效的定义合并，并覆盖所有相同的extension扩展名映射。

如果要指定一个以上的过滤器，它们必须用分号来分隔，并按它们处理文档的顺序来排列。filter与extension参数都是大小无关的，extension可以带或不带前导点。

参见

- `RemoveInputFilter`
- `SetInputFilter`

AddLanguage 指令

说明	在文件扩展名与特定的语言之间建立映射
语法	`AddLanguage MIME-lang extension [extension] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime

`AddLanguage` 指令在文件扩展名与特定的语言之间建立映射。指令定义以extension为扩展名的文件是以MIME-lang语言写成的。这个映射关系会添加在所有有效的映射关系上，并覆盖所有相同的extension扩展名映射。

示例

```
AddEncoding x-compress .Z
AddLanguage en .en
AddLanguage fr .fr
```

这样一来，文档 `xxxx.en.Z` 将会被当成是一个压缩的英文文档(`xxxx.Z.en` 也一样)。虽然内容的语言会返回给客户端，但浏览器一般未必会使用这一信息。 `AddLanguage` 指令更多的用于内容协商，以决定哪个文档应当被返回给用户。

如果同一个扩展名被赋予多个语言，那么使用最后出现的那个。因此在下列情况中：

```
AddLanguage en .en
AddLanguage en-gb .en
AddLanguage en-us .en
```

带 `.en` 扩展名的文档会被当成是 `en-us` 。

extension参数是大小无关的，并且可以带或不带前导点。

参见

- `mod_negotiation`

AddOutputFilter 指令

说明	在文件扩展名与特定的输出过滤器之间建立映射关系
语法	`AddOutputFilter filter[;filter...] extension [extension] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime
兼容性	仅在 Apache 2.0.26 及以后的版本中可用

`AddOutputFilter` 指令在文件扩展名extension与对服务将返回给客户的信息进行处理的输出过滤器之间建立映射。这是除包括 `SetOutputFilter` 和 `AddOutputFilterByType` 指令在内的所有过滤器定义指令之外定义。这个映射会与所有有效的定义合并，并覆盖所有相同的extension扩展名映射。

例如，下述配置会在处理所有 `.shtml` 文件时，进行服务器端包含，并同时使用 `mod_deflate` 模块压缩后输出。

```
AddOutputFilter INCLUDES;DEFLATE shtml
```

如果要指定一个以上的过滤器，它们必须用分号来分隔，并按它们处理文档的顺序来排列。filter和extension参数都是大小写无关的，extension可以带或不带前导点。

参见

- `RemoveOutputFilter`
- `SetOutputFilter`

AddType 指令

说明	在给定的文件扩展名与特定的内容类型之间建立映射
语法	`AddType MIME-type extension [extension] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime

`AddType` 指令在给定的文件扩展名与特定的内容类型之间建立映射关系。MIME-type指明了包含extension扩展名的文件的媒体类型。这个映射关系会添加在所有有效的映射关系上，并覆盖所有相同的extension扩展名映射。本指令可用来增加没有在媒体类型文件(参阅 `TypesConfig` 指令)中定义的映射关系。

示例

```
AddType image/gif .gif
```

推荐使用 `AddType` 指令增加新的媒体类型映射关系，而不是改变 `TypesConfig` 文件。

extension参数是大小无关的，并且可以带或不带前导点。

参见

- `DefaultType`
- `ForceType`

DefaultLanguage 指令

说明	为所有文件设定特定的默认语言
语法	`DefaultLanguage MIME-lang`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime

`DefaultLanguage` 指令告诉Apache，当该指令作用域范围内(例如，所有当前 `<Directory>` 指令封装范围内)的文件没有明确的语言扩展名(例如由 `AddLanguage` 指令定义的 `.fr` 或 `.de`)时，文件应该被认为是由MIME-lang语言构成的。举例来说，这就允许在不必对每个文件进行重命名的情况下，把整个目录中的文件标记为包含荷兰语内容。注意不同于用扩展名来指定语言， `DefaultLanguage` 指令只能指定一种语言。

如果没有提供有效的 `DefaultLanguage` 指令同时文件也不包含由 `AddLanguage` 定义的语言扩展名，那么该文件将被认为没有语言属性。

示例

```
DefaultLanguage en
```

参见

- `mod_negotiation`

ModMimeUsePathInfo 指令

说明	将 **`path_info`** 当成是文件名的一个组成部分
语法	`ModMimeUsePathInfo On|Off`
默认值	`ModMimeUsePathInfo Off`
作用域	directory
状态	基本(B)
模块	mod_mime
兼容性	仅在 Apache 2.0.41 及以后的版本中可用

`ModMimeUsePathInfo` 指令用来设定在使用由 `mod_mime` 提供的指令时，是否将URL的 `path_info` 与文件名结合起来进行处理。默认值为 `Off` ，也就是说URL的 `path_info` 部分被忽略。

当你使用虚拟文件系统的时候，推荐使用本指令。

示例

```
ModMimeUsePathInfo On
```

对 `/bar/foo.shtml` 这样的请求来说，" `/bar` "是一个位置信息，如果 `ModMimeUsePathInfo` 指令为 `On` ， `mod_mime` 会将请求解析成 `/bar/foo.shtml` ， 于是象" `AddOutputFilter INCLUDES .shtml` "这样的指令就会使用 `INCLUDES` 过滤器来处理这个请求。如果没有设定 `ModMimeUsePathInfo` 指令，则不会使用 `INCLUDES` 过滤器。

参见

- `AcceptPathInfo`

MultiviewsMatch 指令

说明	在使用**MultiViews**查询所匹配的文件时要包含的文件类型
语法	`MultiviewsMatch Any|NegotiatedOnly|Filters|Handlers [Handlers|I`
默认值	`MultiviewsMatch NegotiatedOnly`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime
兼容性	仅在 Apache 2.0.26 及以后的版本中可用

`MultiviewsMatch` 指令在实现 `mod_negotiation` 模块的Multiviews功能时，提供了三种不同的处理方法。Multiviews允许对一个文件的请求，可以用任何在基础请求后面跟上协商扩展名的情况来匹配，例如： `index.html` 可以用 `index.html.en` ， `index.html.fr` ， `index.html.gz` 来匹

配。

`NegotiatedOnly` 选项规定每个跟在基础名字后面的扩展名必须关联到一个在内容协商时已知的 `mod_mime` 扩展名，例如：字符集、内容类型、语言、编码方式。这是一种最严格也是副作用最少的实现方法，它是默认的处理方式。

为了包含与处理器和/或过滤器关联的扩展名，可以设定 `MultiviewsMatch` 指令为 `Handlers` 或 `Filters` ，也可以两个都选。如果其它所有因素都相等，则会选择最小的那个文件来提供服务。例如，在一个500字节的 `index.html.cgi` 文件和一个1000字节的 `index.html.pl` 文件中做选择时， `.cgi` 文件会胜出。如果 `.asis` 文件与 `asis-handler` 处理器关联，对 `.asis` 文件的请求就会使用处理器选项指明的处理器。

即使 `mod_mime` 不认识的扩展名，你最终也可以通过使用 `Any` 选项来使它匹配用户的请求。Apaceh1.3就是按这个方式处理的，这会导致无法预测的结果，比如匹配了网站管理员从来不希望使用的.old或.bak文件。

例如，下面的配置允许在Multviews查询中使用处理器和过滤器，但会拒绝未知的文件：

```
MultiviewsMatch Handlers Filters
```

参见

- `Options`
- `mod_negotiation`

RemoveCharset 指令

说明	删除任何给定的扩展名与内容字符集之间的关联
语法	`RemoveCharset extension [extension] ...`
作用域	virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime
兼容性	仅在 Apache 2.0.24 及以后的版本中可用

`RemoveCharset` 指令删除任何给定的扩展名与内容字符集之间的关联。子目录中的 `.htaccess` 文件可以通过这条指令取消从父目录或服务器配置文件中继承过来的扩展名与内容字符集之间的关联关系。

extension参数是大小无关的，并且可以带或不带前导点。

示例

```
RemoveCharset .html .shtml
```

RemoveEncoding 指令

说明	删除任何给定的扩展名与内容编码方式之间的关联
语法	`RemoveEncoding extension [extension] ...`
作用域	virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime

`RemoveEncoding` 指令删除任何给定的扩展名与内容编码方式之间的关联。子目录中的 `.htaccess` 文件可以通过这条指令取消从父目录或服务器配置文件中继承过来的扩展名与内容编码方式之间的关联关系。举例来说，它可以这样来使用：

/foo/.htaccess:

```
AddEncoding x-gzip .gz

AddType text/plain .asc

<Files *.gz.asc>

RemoveEncoding .gz
</Files>
```

这样， `foo.gz` 被认为是用gzip方式编码的，但 `foo.gz.asc` 将被认为是一个未编码的纯文本文件。

注意

`RemoveEncoding` 指令在所有 `AddEncoding` 指令之后处理，因此如果在同一个目录配置里两者都出现的话， `RemoveEncoding` 指令将会取消后面的 `AddEncoding` 指令的作用。

extension参数是大小无关的，并且可以带或不带前导点。

RemoveHandler 指令

说明	删除任何指定扩展名与处理器之间的关联
语法	`RemoveHandler extension [extension] ...`
作用域	virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime

`RemoveHandler` 指令删除任何指定的扩展名与处理器之间的关联。子目录中的 `.htaccess` 文件可以通过这条指令取消从父目录或服务器配置文件中继承过来的扩展名与处理器之间的关联关系。举例来说，它可以这样来使用：

/foo/.htaccess

```
AddHandler server-parsed .html
```

/foo/bar/.htaccess

```
RemoveHandler .html
```

这样 /foo/bar 目录中的 .html 文件将被当成普通文件来处理，而不是由parsing处理器(参阅 mod_include 模块

extension参数是大小无关的，并且可以带或不带前导点。

RemoveInputFilter 指令

说明	删除指定扩展名与输入过滤器之间的关联
语法	`RemoveInputFilter extension [extension] ...`
作用域	virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime
兼容性	仅在 Apache 2.0.26 及以后的版本中可用

`RemoveInputFilter` 指令删除指定的扩展名与输入过滤器之间的关联。子目录中的 `.htaccess` 文件可以通过这条指令取消从父目录或服务器配置文件中继承过来的扩展名与输入过滤器之间的关联关系。

extension参数是大小无关的，并且可以带或不带前导点。

参见

- `AddInputFilter`
- `SetInputFilter`

RemoveLanguage 指令

说明	删除指定的扩展名与内容语言之间的关联
语法	`RemoveLanguage extension [extension] ...`
作用域	virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime
兼容性	仅在 Apache 2.0.24 及以后的版本中可用

`RemoveLanguage` 指令删除指定的扩展名与内容语言之间的关联。子目录中的 `.htaccess` 文件可以通过这条指令取消从父目录或服务器配置文件中继承过来的扩展名与内容语言之间的关联关系。

extension参数是大小无关的，并且可以带或不带前导点。

RemoveOutputFilter 指令

说明	删除指定扩展名与输出过滤器之间的关联
语法	`RemoveOutputFilter extension [extension] ...`
作用域	virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime
兼容性	仅在 2.0.26 及以后的版本中可用

`RemoveOutputFilter` 指令删除指定的扩展名与输出过滤器之间的关联。子目录中的 `.htaccess` 文件可以通过这条指令取消从父目录或服务器配置文件中继承过来的扩展名与输出过滤器之间的关联关系。

extension参数是大小无关的，并且可以带或不带前导点。

示例

```
RemoveOutputFilter shtml
```

参见

- `AddOutputFilter`

RemoveType 指令

说明	删除指定扩展名与内容类型之间的关联
语法	`RemoveType extension [extension] ...`
作用域	virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_mime

`RemoveType` 指令删除指定的扩展名与内容类型之间的关联。子目录中的 `.htaccess` 文件可以通过这条指令取消从父目录或服务器配置文件中继承过来的扩展名与内容类型之间的关联关系。举例来说，它可以这样使用：

/foo/.htaccess

`RemoveType .cgi`

`这将删除` /foo/ `目录及其所有子目录下` .cgi `文件的特定处理方式，从而使这些文件按` DefaultType `指令设定的默认`

`

注意

`RemoveType` 指令会在所有的 `AddType` 指令之后处理，因此，当同一个目录配置中，同时存在这两种指令的时候，前面的 `RemoveType` 指令将会取消后面的 `AddType` 指令的作用。

extension参数是大小无关的，并且可以带或不带前导点。

``

TypesConfig 指令

说明	指定 `mime.types` 文件的位置
语法	`TypesConfig file-path`
默认值	`TypesConfig conf/mime.types`
作用域	server config
状态	基本(B)
模块	mod_mime

`TypesConfig` 指令设定MIME类型配置文件的位置。File-path是相对于 `ServerRoot` 的路径。媒体类型配置文件列出了文件扩展名与内容类型的默认映射关系。大多数管理员使用既定的 `mime.types` 文件，它关联了文件扩展名和由IANA注册的内容类型。最新的列表可以在http://www.iana.org/assignments/media-types/index.html得到。这样做可以大大简化 `httpd.conf` 文件里的媒体类型定义，在需要时，也可以用 `AddType` 指令来更改这些定义。你不应该编辑 `mime.types` 文件，因为在服务器升级的时候，它会被覆盖。

文件包含类似于 `AddType` 指令参数格式的行：

```
<var class="calibre40">MIME-type</var> [<var class="calibre40">extension</var>] ...
```

扩展名的大小写是无关紧要的。空行和以井号(`#`)打头的行会被忽略。

请不要要求Apache HTTP服务器项目组在已发布的 `mime.types` 文件中增加新的项，除非(1)它们已经在IANS注册过了，或者(2)它们被广泛地使用，并且在多平台上没有文件扩展名冲突发生。 `category/x-subtype` 请求会被自动拒绝，因为任何新的二字母的扩展名很可能会与已经非常拥挤的语言及字符集名字空间冲突。

参见

- `mod_mime_magic`

``

Apache模块 mod_mime_magic

说明	通过读取部分文件内容自动猜测文件的MIME类型
状态	扩展(E)
模块名	mime_magic_module
源文件	mod_mime_magic.c

概述

本模块采取Unix系统下 `file(1)` 命令相同的方法：检查文件开始的几个字节，来判定文件的MIME类型。它被作为当 `mod_mime` 无法解析时，用来处理的"第二道防线"。

本模块源自于Unix系统命令 `file(1)` 的一个自由版本，它通过对来自文件的内容使用"Magic数字"和其它一些线索来判定这个文件的具体内容是什么。本模块只有当"Magic文件"在 `MimeMagicFile` 指令中指定时才有效。

"Magic文件"的格式

Magic文件的内容是由4-5列的纯文本组成的。文件中允许包含空行，但会被忽略。注释行使用井号(`#`)来引导。剩余的行按下面列被分解：

列	描述
1	开始检查的起始字节，">"表示基于前面的非">"行。
2	匹配的数据类型 `byte` 单个字符 `short` 机器字节顺序的16位整数 `long` 机器字节顺序的32位整数 `string` 任意长度的字符串 `date` 长整型的日期(从UNIX纪元/1970以来的秒数) `beshort` big-endian 16位整数 `belong` big-endian 32位整数 `bedate` big-endian 32位整型日期 `leshort` little-endian 16位整数 `lelong` little-endian 32位整数 `ledate` little-endian 32位整型日期
3	匹配的数据内容
4	如果匹配文件的MIME类型
5	如果匹配文件的MIME编码方式(可选)

例如，下面的Magic文件行可以认出一些音频格式：

```
# Sun/NeXT audio data
0       string      .snd
>12     belong      1       audio/basic
>12     belong      2       audio/basic
>12     belong      3       audio/basic
>12     belong      4       audio/basic
>12     belong      5       audio/basic
>12     belong      6       audio/basic
>12     belong      7       audio/basic
>12     belong     23       audio/x-adpcm
```

还有下面的示例可以区分带 `*.doc` 扩展名的文件到底是Microsoft Word文档还是Frame Maker文档(两种有相同后缀名但不兼容文件格式)。

```
# Frame
0  string  \<MakerFile        application/x-frame
0  string  \<MIFFile          application/x-frame
0  string  \<MakerDictionary  application/x-frame
0  string  \<MakerScreenFon   application/x-frame
0  string  \<MML              application/x-frame
0  string  \<Book             application/x-frame
0  string  \<Maker            application/x-frame

# MS-Word
0  string  \376\067\0\043            application/msword
0  string  \320\317\021\340\241\261  application/msword
0  string  \333\245-\0\0\0           application/msword
```

一个可选的MIME编码方式可以包含在第五列上。例如下面的行可以认出gzip压缩文件并设定他们的编码方式。

```
# gzip (GNU zip, not to be confused with
#       [Info-ZIP/PKWARE] zip archiver)

0  string  \037\213  application/octet-stream  x-gzip
```

性能问题

并不是每个系统都适用本模块的。如果你的系统吞吐量已经接近极限，或者你在进行web服务器的基准测试，你可能不希望启动这个模块，因为它的处理会显著影响服务器的性能。

然而，已经有人在努力改进最初的 `file(1)` 代码，使它能更适合在一个非常繁忙的web服务器上工作。这主要是用在那种有数千用户自己发布文档的web服务器上。这在互联网上可能是非常常见的情况。很多情况下，如果服务器能就文件的内容作出比用文件名来区别的方式更加智能化的判断是非常有用的。甚至在当用户没有合理地命名他们的文件的情况下，它也可以用来减少那些诸如："为什么我的页面不工作啊"之类的报怨。你必须自己决定这额外的开销是否适用于你的环境。

注意

下面关于 `mod_mime_magic` 的记录包含在这里，是按照最初捐助者的版权限制和为了得到他们的承认。

mod_mime_magic: 通过文件的Magic Number查找文件的MIME类型 Copyright (c) 1996-1997 Cisco Systems, Inc.

本软件由Cisco系统有限公司于1997年7月提交给Apache组织。本软件源代码的进一步修订及新版本的派生必须承认Cisco系统有限公司是本模块的原始捐助者。所有其它许可与使用条件都属于Apache组织。

本模块的部分源代码派生于最初发布在comp.sources.unix上的file命令的自由版本。根据要求，下面包含了那个程序的版权信息。

- Copyright (c) Ian F. Darwin, 1987. Written by Ian F. Darwin.

本软件不隶属于美国电话电报公司(AT&T)或者加利福尼亚大学董事会的任何许可。

在遵循以下限制的基础上，任何人被授权免费地在任何计算机系统上为任何目的使用、修改与重新分发这个软件：

1. 不管后果有多严重，甚至是直接由于程序的缺陷造成的，程序的作者对于由于使用这个软件而造成的任何直接或间接的后果不承担任何责任。
2. 在清晰或冗长的声明中，软件的来源必须无误地叙述。由于少量用户可能会阅读源代码，源代码中也必须包含这一声明。
3. 改动版本必须明白标明，必须与原软件严格区分开来。由于少量用户可能会阅读源代码，源代码中也必须包含这一声明。
4. 本声明不能被删除或更改。

为了符合MrDarwin的条款：这是从自由的"file"命令而来并经过了明显的修改的版本。

- 当从Apache的一个版本转移到下一个时，为了编辑方便，所有代码在一个文件中。
- 内存分配通过Apache应用程序接口的缓冲池结构完成。
- 在需要调用其它Apache应用程序接口例程的地方，所有的函数被提供必需的Apache应用程序接口及服务结构。(例如：通常在它自身或被调用的程序里包含了日志记录，文件操作或内存分配)
- Magic结构从数组被转换成了单终点链表，因为它每次只增长一条记录，它只按顺序方式处理，同时Apache API没有realloc()的替代处理方法。
- 函数被改变成从参数获取服务器配置，而不是全局变量。(现在它应该是可重入的，但没有在线程化的环境中测试过)
- 原来用来打印结果到stdout的地方，被改成将结果保存到一个列表，这个列表被用来在Apache请求记录中设置MIME类型。
- 因为在这里再也不会用到命令行标志，它们被删除了。

MimeMagicFile 指令

说明	使用特定的**Magic**文件激活根据文件内容确定文件**MIME**类型的功能
语法	`MimeMagicFile file-path`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_mime_magic

`MimeMagicFile` 指令用来激活本模块，默认的Magic文件保存在 `conf/magic` 中。相对路径是相对于 `ServerRoot` 的。虚拟主机会使用与主服务器相同的配置文件，除非使用了更特别的设定。在后者情况下，这些特别的设定会覆盖主服务的设定。

示例

```
MimeMagicFile conf/magic
```

Apache模块 mod_negotiation

说明	提供内容协商支持
状态	基本(B)
模块名	negotiation_module
源文件	mod_negotiation.c

概述

内容协商，更准确的说应该是"内容选择"，是从几个有效文档中选择一个最匹配客户端要求的文档的过程。内容协商有两种实现方法。

- 使用类型表(一个包含 `type-map` 处理器信息的文件)明确地列出各变种的文件名。
- 使用"MultiViews"搜索(由 `Options` 指令中的 `MultiViews` 选项激活)，即服务器执行一个隐含的文件名模式匹配，并在结果中选择。

类型表

类型表的格式有点类似于RFC822邮件头的格式。它包含以空行分格的文档描述，以井号(#)打头的行被当作是注释。一个文档描述包含几个头记录，以空格开始的行将被认为是前一行的延续，这样文档描述记录就可以包含多行。在处理处理多行记录时，行与行被连接起来，打头的空格会被删除。一个头记录包含一个关键字名，并且总是在结尾的地方用一个冒号将它自己和紧跟其后的值分隔开。在头名字与它的值之间以及取值的各个标记之间可以插入空格。头可以是：

`Content-Encoding:`

文件的编码方式。Apache只支持用 `AddEncoding` 指令定义的编码方式。它一般包含compress压缩文件的 `x-compress` 编码和gzip文件的 `x-gzip` 编码。在编码对照过程中，" `x-` "前缀会被忽略。

`Content-Language:`

按互联网标准语言标签(RFC 1766)定义的变体语言。举例来说 `en` 表示英语。如果变体包含一种以上的语言，用逗号来分隔。

`Content-Length:`

以字节为单位的文件长度。如果这个头不存在，则使用文件的实际长度。

`Content-Type:`

带可选参数的文档MIME类型。参数与MIME类型之间以及参数之间都用分号分隔，参数使用类似" `name=value` "这样的语法。参数包括：

`level`

一个指明媒体类型版本的整数。对 `text/html` 来讲，默认值是"2"，其它的默认值为"0"。

`qs`

一个取值在0.0到1.0之间的浮点数，它表示在不考虑客户端性能的情况下，一个变体相对于其它变体的"品质"。比如在表现一张照片时，jpeg文件通常比字符构图有较高的还原品质；而如果要表现的本来就是一个字符构图，那么当然Ascii文件会比jpeg文件有较高的还原品质。因此，所有的 `qs` 取值都是特定于某个资源的。

示例

```
Content-Type: image/jpeg; qs=0.8
```

`URI:`

文件的URI包含了媒体类型、编码方式等变量的信息，这些被解释为与映射文件相关的URL，它们必须在同一个服务器上。如果它们被直接调用的话，它们所涉及的文件必须对用户是可以访问的。

`Body:`

这是2.0新增的功能，使用Body头，资源的实际内容可以直接包含在类型表里。这个头必须包含一个指明分隔符的字符串。这样在类型表文件中，接下来直到分隔字符串之前的所有内容，会被当作是资源实体。

示例：

```
Body:----xyz----
&lt;html&gt;
&lt;body&gt;
&lt;p&gt;Content of the page.&lt;/p&gt;
&lt;/body&gt;
&lt;/html&gt;
----xyz----
```

MultiViews

MultiViews查询是由 `Options` 指令的 `MultiViews` 选项激活的。如果服务器接收了一个对 `/some/dir/foo` 的请求，而 `/some/dir/foo` 并不存在，则服务器会查找这个目录下所有的 `foo.*` 文件，并有效地伪造一个说明这些 `foo.*` 文件的类型表，假定客户可能请求的一个，把他们指定为这个类型的媒体类型及内容编码。最终选择其中最符合客户请求的文档，返回给客户。

`MultiViewsMatch` 指令指示Apache在选择文件时是否考虑不包含内容协商元信息的文件。

CacheNegotiatedDocs 指令

说明	允许经过内容协商的文档被代理服务器缓存
语法	`CacheNegotiatedDocs On|Off`
默认值	`CacheNegotiatedDocs Off`
作用域	server config, virtual host
状态	基本(B)
模块	mod_negotiation
兼容性	从2.0版本起，语法有了变化

如果设置为"On"，则允许内容协商文档被代理服务器缓存。这可能意味着在代理服务器后面的客户端得到的文档并不是最符合他们情况的版本，但它能够使缓存更有效。

本指令只对HTTP/1.0浏览器的请求有效。HTTP/1.1在对内容协商文档进行缓冲方面提供了更好的控制，本指令对于HTTP/1.1的应答没有效果。

在2.0版本以前， `CacheNegotiatedDocs` 指令不带任何参数；它会自己在目录存在的情况下开启。

ForceLanguagePriority 指令

说明	指定无法匹配单个文档的情况下所采取的动作
语法	`ForceLanguagePriority None|Prefer|Fallback [Prefer|Fallback]`
默认值	`ForceLanguagePriority Prefer`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_negotiation
兼容性	仅在 Apache 2.0.30 及以后的版本中可用

`ForceLanguagePriority` 指令使用 `LanguagePriority` 指令的设置，在服务器无法返回单个匹配文档的情况下，指定完成协商过程的方法。

`ForceLanguagePriority Prefer` 在有几个等价选择的情况下，使用 `LanguagePriority` 的设定以提供一个有效的结果，而不是返回HTTP结果"300"(多重选择)。如果给出了下述指令，并且用户的 `Accept-Language` 头为 `en` 和 `de` 赋予了相同的品质系数" `.500` "(相同的品质系数是允许的)，那么第一个匹配的变体： `en` 将被返回。

```
LanguagePriority en fr de
ForceLanguagePriority Prefer
```

`ForceLanguagePriority Fallback` 使用 `LanguagePriority` 指令在无法找到合适结果的情况下，指定一个有效的结果，而不是返回HTTP结果"406"(不可接受)。如果给出了下述指令，并且用户的 `Accept-Language` 头只允许 `es` 的返回结果，在这个变体没有找到的情况下，下述 `LanguagePriority` 指令列表的第一个变体将被返回。

```
LanguagePriority en fr de
ForceLanguagePriority Fallback
```

`Prefer` 和 `Fallback` 两个选项可以同时指定，这样在有一个以上有效变体的情况下，返回 `LanguagePriority` 指令列表中第一个匹配的变体文档，而在没有变体能够匹配客户可接受的语言的情况下，返回第一个可用的变体文档。

参见

- `AddLanguage`

LanguagePriority 指令

说明	在客户端没有指示语言偏好的情况下，语言变体的优先级列表
语法	`LanguagePriority MIME-lang [MIME-lang] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_negotiation

在处理MultiViews请求时， `LanguagePriority` 指令在客户没有指示语言偏爱的情况下，设定语言变体的优先级列表。这个MIME-lang列表是按优先级降序排列的。

示例：

```
LanguagePriority en fr de
```

表示对于 `foo.html` 请求，如果 `foo.html.fr` 和 `foo.html.de` 同时存在，并且浏览器没有表明对语言的偏爱，那么 `foo.html.fr` 将被返回。

注意，本指令只在根据其它信息无法决定最好的语言或者 `ForceLanguagePriority` 指令不是 `None` 时才有效。对正确实现的HTTP/1.1请求，本指令没有任何作用。

参见

- `AddLanguage`

Apache模块 mod_nw_ssl

说明	仅限于在**NetWare**平台上实现**SSL**加密支持
状态	基本(B)
模块名	nwssl_module
源文件	mod_nw_ssl.c
兼容性	仅用于NetWare

概述

本模块在指定的端口(port)上启用SSL加密。它充分利用了NetWare操作系统内建的SSL加密功能。

NWSSLTrustedCerts 指令

说明	附加的客户端证书列表
语法	`NWSSLTrustedCerts filename [filename] ...`
作用域	server config
状态	基本(B)
模块	mod_nw_ssl

指定客户证书文件(DER格式)列表，这些文件用于创建代理SSL连接。每个客户证书必须各自列出自己的 `.der` 文件。

NWSSLUpgradeable 指令

说明	允许在请求中将一个连接升级为**SSL**连接
语法	`NWSSLUpgradeable [IP-address:]portnumber`
作用域	server config
状态	基本(B)
模块	mod_nw_ssl

允许客户端在请求中将特定地址和/或端口上的连接升级为SSL连接。这个地址和/或端口必须先前已经使用 `Listen` 指令定义过。

SecureListen 指令

说明	在指定端口启用SSL加密
语法	`SecureListen [IP-address:]portnumber Certificate-Name [MUTUAL]`
作用域	server config
状态	基本(B)
模块	mod_nw_ssl

指定端口和启用SSL加密时需要用到的基于eDirectory的证书名称。第三个可选的参数启用mutual认证。

Apache模块 mod_proxy

说明	提供HTTP/1.1的代理/网关功能支持
状态	扩展(E)
模块名	proxy_module
源文件	mod_proxy.c

概述

警告

在您没有对服务器采取安全措施之前，请不要用 `ProxyRequests` 启用代理。一个开放的代理服务器不仅对您的网络有威胁，对整个因特网来说也同样如此。

此模块实现了Apache的代理/网关。它实现了以下规范的代理： `AJP13` (Apache JServe Protocol v1.3), `FTP` , `CONNECT` (用于SSL), `HTTP/0.9` , `HTTP/1.0` , `HTTP/1.1` 。此模块经配置后可用上述或其它协议连接其它代理模块。

Apache的代理功能(除 `mod_proxy` 以外)被划分到了几个不同的模块中： `mod_proxy_http` , `mod_proxy_ftp` , `mod_proxy_ajp` , `mod_proxy_balancer` , `mod_proxy_connect` 。这样，如果想使用一个或多个代理功能，就必须将 `mod_proxy` 和对应的模块同时加载到服务器中(静态连接或用 `LoadModule` 动态加载)。

另外，其它模块还提供了扩展特性。 `mod_cache` 及其相关模块提供了缓冲特性。 `mod_ssl` 提供的 `SSLProxy*` 系列指令可以使用SSL/TLS连接远程服务器。这些提供扩展特性的模块必须在被正确加载和配置以后才能提供这些扩展功能。

正向和反向代理

Apache可以被配置为<dfn class="calibre27">正向(forward)</dfn>和<dfn class="calibre27">反向(reverse)</dfn>代理。

<dfn class="calibre27">正向代理</dfn>是一个位于客户端和原始服务器*(origin server)*之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。客户端必须要进行一些特别的设置才能使用正向代理。

正向代理的典型用途是为在防火墙内的局域网客户端提供访问Internet的途径。正向代理还可以使用缓冲特性(由 `mod_cache` 提供)减少网络使用率。

使用 `ProxyRequests` 指令即可激活正向代理。因为正向代理允许客户端通过它访问任意网站并且隐藏客户端自身，因此你必须采取安全措施以确保仅为经过授权的客户端提供服务。

<dfn class="calibre27">反向代理</dfn>正好相反，对于客户端而言它就像是原始服务器，并且客户端不需要进行任何特别的设置。客户端向反向代理的名字空间(name-space)中的内容发送普通请求，接着反向代理将判断向何处(原始服务器)转交请求，并将获得的内容返回给客户端，就像这些内容原本就是它自己的一样。

反向代理的典型用途是将防火墙后面的服务器提供给Internet用户访问。反向代理还可以为后端的多台服务器提供负载平衡，或为后端较慢的服务器提供缓冲服务。另外，还可以启用高级URL策略和管理技术，从而使处于不同web服务器系统的web页面同时存在于同一个URL空间下。

可以使用 `ProxyPass` 指令激活反向代理(在 `RewriteRule` 指令中使用 `P]` 标记也可以)。配置反向代理并不需要打开 `[ProxyRequests` 指令。

简单示例

下面的例子仅仅是为了给你一个基本概念而帮助入门而已，请仔细阅读每个指令的文档。

另外，如果想使用缓冲特性，请查看 `mod_cache` 文档。

正向代理

```
    ProxyRequests On

    ProxyVia On

    <Proxy *>

 Order deny,allow

      Deny from all

      Allow from internal.example.com
    </Proxy>
```

反向代理

```
    ProxyRequests Off

    <Proxy *>

Order deny,allow

      Allow from all
    </Proxy>

    ProxyPass /foo http://foo.example.com/bar

    ProxyPassReverse /foo http://foo.example.com/bar
```

控制对代理服务器的访问

您可以通过 `<Proxy>` 的阻止功能来控制谁能访问您的代理。示例如下：

```
        <Proxy *>

Order Deny,Allow

          Deny from all

          Allow from 192.168.0
        </Proxy>
```

要了解更多访问控制信息，请参见 `mod_authz_host` 文档。

使用正向代理时严格控制访问权限(使用 `ProxyRequests` 指令)是非常重要的。否则你的代理会被客户端利用来访问其它服务器并且隐藏客户端的真实身份。这不仅对您的网络有威胁，对整个因特网来说也同样如此。当使用反向代理(在" `ProxyRequests Off` "条件下使用 `ProxyPass` 指令)的时候访问控制要相对宽松，因为客户端只能连接你配置的特定主机。

缓慢启动

如果您使用了 `ProxyBlock` 指令，将会在启动时查找并缓存主机名的IP地址以备后继的匹配测试使用。这将会花费几秒或更长的时间，这主要取决于主机名查找的速度。

局域网代理

位于局域网内的Apache代理服务器需要经由公司的防火墙转发对外部的请求(使用 `ProxyRemote` 指令来配置)。但当它访问局域网内的资源时，它能越过防火墙直接访问目的主机。在访问一个属于局域网的服务器从而进行直接连接时， `NoProxy` 指令就会很有用。

局域网内的用户习惯于不在他们的WWW请求中加入本地域的名称，于是会使用"http://somehost/"来取代`http://somehost.example.com/` 。一些商业代理服务器会不管这些，只是采用本地域的配置来简单的伺服这个请求。当使用了 `ProxyDomain` 指令来为服务器配

置了一个代理服务时，Apache会发出一个重定向应答，以使客户端请求到达正确的、能满足要求的服务器地址。因为这样一来，用户的书签文件就会随之包含完整的主机名，所以这是首选的方法。

协议调整

当 `mod_proxy` 向一个没有正确实现持久连接(KeepAlive)或HTTP/1.1的原始服务器发送请求的时候，可以通过设置两个环境变量来发送不带持久连接(KeepAlive)的HTTP/1.0请求。这两个变量是通过 `SetEnv` 指令设置的。

以下是 `force-proxy-request-1.0` 和 `proxy-nokeepalive` 的例子：

```
        <Location /buggyappserver/>
 ProxyPass http://buggyappserver:7001/foo/
          SetEnv force-proxy-request-1.0 1
          SetEnv proxy-nokeepalive 1
        </Location>
```

请求体

一些请求方法(如POST)包含一个请求体。HTTP协议要求包含请求体的请求或者使用块传输编码(chunked transfer encoding)或者包含一个 `Content-Length` 请求头。当将这种请求传递给原始服务器的时候， `mod_proxy_http` 会始终尝试使用 `Content-Length` 请求头。但如果原始请求使用的是块编码，那么块编码也同样可以用于上行请求。可以使用环境变量控制这种选择。设置 `proxy-sendcl` 可以确保始终发送 `Content-Length` 头以与上游服务器保持最大程度的兼容性，而设置 `proxy-sendchunked` 可以通过继续使用块编码以尽可能最小化资源占用率。

AllowCONNECT 指令

说明	通过代理允许 `CONNECT` 的端口号
语法	`AllowCONNECT port [port] ...`
默认值	`AllowCONNECT 443 563`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

`AllowCONNECT` 指令指定了此代理的 `CONNECT` 方法可以连接的端口号列表。当今的浏览器在进行 `https` 连接请求时使用这种方法，而代理默认会将其转为 `http` 。

默认只启用了默认的https端口(`443`)和默认的snews端口(`563`)。使用`AllowCONNECT`指令可以覆盖默认设置而改为仅允许连接列出的端口。

注意，必须确保`mod_proxy_connect`也同时存在于服务器中，这样才能支持`CONNECT`。

NoProxy 指令

说明	直接进行连接的主机/域/网络
语法	`NoProxy host [host] ...`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

此指令仅适用于局域网内的Apache代理服务器。`NoProxy`指令指定了一个中间以空格分隔的子网、IP地址、主机和/或域的列表。对某个匹配上述一个或多个列表项的主机的请求将直接被其伺服而不会转交到配置好的`ProxyRemote`代理服务器。

示例

```
ProxyRemote  *  http://firewall.mycompany.com:81
NoProxy         .mycompany.com 192.168.112.0/21
```

`NoProxy`指令的host参数可以是以下选项之一：

域

<dfn class="calibre27">域</dfn>是一个DNS域名的一部分，并在前面加上点号。它表示一批逻辑上属于同一个DNS区域的主机，也就是所有这些主机名具有相同的后缀，而这个"后缀"就是<dfn class="calibre27">域</dfn>。

例子

```
    .com
.apache.org.
```

域和主机名(一个DNS域甚至也可能有一条DNS"A记录"!)的不同之处在于域始终有一个前导点。

注意

域名不区分大小写并且始终认为是锚定在DNS树根上的，因此 `.MyDomain.com` 和 `.mydomain.com.` (注意结尾点号)是完全等同的。因为域的比较不需要进行DNS查询，因此它比子网比较更加高效。

子网

<dfn class="calibre27">子网</dfn>以点分十进制形式表示了一个因特网地址的一部分，有时会跟着一个斜杠和子网掩码，以指定子网中的有效bit位。它用于表示主机通过自身的普通网络接口可以访问的子网范围。未指定子网掩码的时候就假定忽略掉的(或为零的)结尾数字就是掩码，在这种情况下，掩码bit长度必须是8bit的整数倍。例如：

`192.168` 或 `192.168.0.0`

子网" `192.168.0.0` "表示掩码为16bit(有时也用 `255.255.0.0` 表示)。

`192.168.112.0/21`

子网" `192.168.112.0/21` "表示掩码为21bit(有时也用 `255.255.248.0` 表示)。

在退化到极限的情况下，一个掩码为32bit的子网就等价于一个IP地址。而零个合法bit的子网("0.0.0.0/0")等价于常量"*Default*"，可以匹配任何IP地址。

IP地址

<dfn class="calibre27">IP地址</dfn>以点分十进制形式表示了一个完整的因特网地址。一般来说，此地址代表一个主机，但并不需要一个DNS域名与这个地址对应。

示例

```
192.168.123.7
```

注意

一个IP地址不需要为一个DNS系统所解析，所以它能使apache获取更高性能。

主机名

<dfn class="calibre27">主机名</dfn>是一个完整的DNS域名，可以通过DNS域名服务解析为一个或多个IP地址。它代表了一个逻辑主机(与域相反)而且必须解析成至少一个IP地址(或经常解析成具有不同IP地址的主机列表)。

例子

```
prep.ai.mit.edu

www.apache.org
```

注意

在很多情况下，指定一个IP地址代替主机名会更有效率。因为可以避免一次DNS查询。当使用一个低速的PPP与域名服务器连接时，Apache的域名解析会花费相当可观的时间。

主机名不区分大小写并且始终认为是锚定在DNS树根上的，因此 `WWW.MyDomain.com` 和 `www.mydomain.com.` (注意结尾点号)是完全等同的。

参见

- DNS相关问题

<Proxy> 指令

说明	应用于所代理资源的容器
语法	`<Proxy wildcard-url> ...</Proxy>`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

位于 `<Proxy>` 配置段中的指令仅作用于匹配的代理内容。语句中可以使用shell风格的通配符。

比如说：下例仅允许 `yournetwork.example.com` 中的主机通过您的代理服务器访问代理内容：

```
     <Proxy *>
Order Deny,Allow
       Deny from all
       Allow from yournetwork.example.com
     </Proxy>
```

下例将在所有 `example.com` 的 `foo` 目录下的文件通过代理服务器发送之前用 `INCLUDES` 过滤器进行处理：

```
     <Proxy http://example.com/foo/*>
SetOutputFilter INCLUDES
     </Proxy>
```

ProxyBadHeader 指令

说明	确定如何处理不合法的应答头
语法	`ProxyBadHeader IsError|Ignore|StartBody`
默认值	`ProxyBadHeader IsError`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy
兼容性	仅在 Apache 2.0.44 及以后的版本中可用

`ProxyBadHeader` 指令决定 `mod_proxy` 如何处理不合法的应答头(比如丢失冒号(:))。参数的取值范围如下：

`IsError`

以"502"(Bad Gateway)应答中止请求。这是默认行为。

`Ignore`

忽略，就像它们不存在一样。

`StartBody`

在接收到第一个非法头行时停止读取头，并将剩余部分当作应答体。这样做有助于和一个不规范的、经常忘记在应答头和应答体之间插入空行的后端服务器协同工作。

ProxyBlock 指令

说明	设置被代理屏蔽的语句、主机、域
语法	`ProxyBlock *|word|host|domain [word|host|domain] ...`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

`ProxyBlock` 指令指定了一个由空格分隔的语句、主机和/或域的列表。对所有匹配这些语句、主机和/或域的HTTP、HTTPS、FTP文档的请求都将被代理服务器阻断。代理模块亦会在启动时尝试确定列表中可能是主机名的项目对应的IP地址，并将其缓冲用于匹配测试。比如说：

示例

```
ProxyBlock joes-garage.com some-host.co.uk rocky.wotsamattau.edu
```

通过IP地址， `rocky.wotsamattau.edu` 将可能同样被匹配。

请注意， `wotsamattau` 已经足够匹配 `wotsamattau.edu` 了。

请注意

```
ProxyBlock *
```

将屏蔽对所有站点的连接。

ProxyDomain 指令

说明	代理请求的默认域名
语法	`ProxyDomain Domain`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

此指令仅对位于局域网内的Apache代理服务器有用。 `ProxyDomain` 指令指定了apache代理服务器归属的默认域。如果遇到了一个对没有域名的主机的请求，就会根据配置自动生成一个加上了Domain的重定向应答。

示例

```
ProxyRemote  *  http://firewall.mycompany.com:81
NoProxy         .mycompany.com 192.168.112.0/21
ProxyDomain     .mycompany.com
```

ProxyErrorOverride 指令

说明	覆盖代理内容的错误页
语法	`ProxyErrorOverride On|Off`
默认值	`ProxyErrorOverride Off`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy
兼容性	仅在 Apache 2.0 及以后的版本中可用

此指令用于反向代理设置中您想为最终用户提供观感一致的错误页面时。它也同样允许包含文件(通过 `mod_include` 的SSI)获取错误号并作出相应的动作。(默认行为是显示被代理的服务器的错误页面，将此项目设为"On"将显示SSI错误信息。)

ProxyIOBufferSize 指令

说明	内部缓冲区大小
语法	`ProxyIOBufferSize bytes`
默认值	`ProxyIOBufferSize 8192`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

`ProxyIOBufferSize` 指令用于调整内部缓冲区(作为输入输出数据的暂存器)的大小。取值必须小于等于 `8192` 。

在绝大多数情况下，不需要调整这个设置。

<ProxyMatch> 指令

说明	应用于匹配正则表达式的代理资源的容器
语法	`<ProxyMatch regex> ...</ProxyMatch>`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

`<ProxyMatch>` 和 `<Proxy>` 指令基本相同，只是匹配字符串可以为正则表达式。

ProxyMaxForwards 指令

说明	转发请求的最大代理数目
语法	`ProxyMaxForwards number`
默认值	`ProxyMaxForwards 10`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy
兼容性	仅在 Apache 2.0 及以后的版本中可用

`ProxyMaxForwards` 指令指定了允许转发请求的最大代理数目。这个设置是为了避免无限代理循环或DoS攻击的发生。

示例

```
ProxyMaxForwards 15
```

ProxyPass 指令

说明	将一个远端服务器映射到本地服务器的URL空间中
语法	`ProxyPass [path] !|url [key=value key=value ...]]`
作用域	server config, virtual host, directory
状态	扩展(E)
模块	mod_proxy

该指令允许你将一个远端服务器映射到本地服务器的URL空间中，此时本地服务器并不充当代理角色，而是充当远程服务器的一个镜像。path是一个本地虚拟路径名，url是一个指向远程服务器的部分URL，并且不允许包含查询字符串。

当使用 `ProxyPass` 指令时， `ProxyRequests` 指令通常应当被设为 **off** 。

假设本地服务器地址是： `http://example.com/` ， 那么，

```
ProxyPass /mirror/foo/ http://backend.example.com/
```

将会导致对 `http://example.com/mirror/foo/bar` 的本地请求将会在内部转换为一个代理请求： `http://backend.example.com/bar` 。

" `!` "指令对于您不想对某个子目录进行反向代理时很有用。比如说：

```
ProxyPass /mirror/foo/i !
ProxyPass /mirror/foo http://backend.example.com
```

将会代理除 `/mirror/foo/i` 之外的所有对 `backend.example.com` 下 `/mirror/foo` 的请求。

注意

顺序很重要，您需要把拒绝指令放置在普通 `ProxyPass` 指令之前。

As of Apache 2.1, the ability to use pooled connections to a backend server is available. Using the `key=value` parameters it is possible to tune this connection pooling. The default for a `Hard Maximum` for the number of connections is the number of threads per process in the active MPM. In the Prefork MPM, this is always 1, while with the Worker MPM it is controlled by the `ThreadsPerChild` .

Setting `min` will determine how many connections will always be open to the backend server. Upto the Soft Maximum or `smax` number of connections will be created on demand. Any connections above `smax` are subject to a time to live or `ttl` . Apache will never create more than the Hard Maximum or `max` connections to the backend server.

```
ProxyPass /example http://backend.example.com smax=5 max=20 ttl=120 retry=300
```

Parameter	Default	Description
min	0	Minumum number of connections that will always be open to the backend server.
max	1...n	Hard Maximum number of connections that will be allowed to the backend server. The default for a Hard Maximum for the number of connections is the number of threads per process in the active MPM. In the Prefork MPM, this is always 1, while with the Worker MPM it is controlled by the `ThreadsPerChild` . Apache will never create more than the Hard Maximum connections to the backend server.
smax	max	Upto the Soft Maximum number of connections will be created on demand. Any connections above `smax` are subject to a time to live or `ttl` .
ttl	-	Time To Live for the inactive connections above the `smax` connections in seconds. Apache will close all connections that has not been used inside that time period.
timeout	`Timeout`	Connection timeout in seconds. If not set the Apache will wait until the free connection is available. This directive is used for limiting the number of connections to the backend server together with `max` parameter.

acquire	-	If set this will be the maximum time to wait for a free connection in the connection pool. If there are no free connections in the pool the Apache will return `SERVER_BUSY` status to the client.
keepalive	Off	This parameter should be used when you have a firewall between your Apache and the backend server, who tend to drop inactive connections. This flag will tell the Operating System to send `KEEP_ALIVE` messages on inactive connections (interval depends on global OS settings, generally 120ms), and thus prevent the firewall to drop the connection. To enable keepalive set this property value to `On` .
retry	60	Connection pool worker retry timeout in seconds. If the connection pool worker to the backend server is in the error state, Apache will not forward any requests to that server until the timeout expires. This enables to shut down the backend server for maintenance, and bring it back online later.
loadfactor	1	Worker load factor. Used with BalancerMember. It is a number between 1 and 100 and defines the normalized weighted load applied to the worker.
route	-	Route of the worker when used inside load balancer. The route is a value appended to seesion id.
redirect	-	Redirection Route of the worker. This value is usually set dynamically to enable safe removal of the node from the cluster. If set all requests without session id will be redirected to the BalancerMember that has route parametar equal as this value.

If the Proxy directive scheme starts with the `balancer://` then a virtual worker that does not really communicate with the backend server will be created. Instead it is responsible for the management of several "real" workers. In that case the special set of parameters can be add to this virtual worker.

Parameter	Default	Description
lbmethod	-	Balancer load-balance method. Select the load-balancing scheduler method to use. Either `byrequests` , to perform weighted request counting or `bytraffic` , to perform weighted traffic byte count balancing. Default is `byrequests` .
stickysession	-	Balancer sticky session name. The value is usually set to something like `JSESSIONID` 或 `PHPSESSIONID` , and it depends on the backend application server that support sessions.
nofailover	Off	If set to `On` the session will break if the worker is in error state or disabled. Set this value to On if backend servers do not support session replication.
timeout	0	Balancer timeout in seconds. If set this will be the maximum time to wait for a free worker. Default is not to wait.
maxattempts	1	Maximum number of failover attempts before giving up.

```
      ProxyPass /special-area http://special.example.com/ smax=5 max=10

      ProxyPass / balancer://mycluster stickysession=jsessionid nofailover=On

      <Proxy balancer://mycluster>

 BalancerMember http://1.2.3.4:8009

        BalancerMember http://1.2.3.5:8009 smax=10

        # Less powerful server, don't send as many requests there

        BalancerMember http://1.2.3.6:8009 smax=1 loadfactor=20
      </Proxy>
```

When used inside a `<Location>` section, the first argument is omitted and the local directory is obtained from the `<Location>` .

If you require a more flexible reverse-proxy configuration, see the `RewriteRule` directive with the `[P]` flag.

ProxyPassReverse 指令

说明	调整由反向代理服务器发送的HTTP应答头中的URL
语法	`ProxyPassReverse [path] url`
作用域	server config, virtual host, directory
状态	扩展(E)
模块	mod_proxy

此指令使Apache调整HTTP重定向应答中 `Location` , `Content-Location` , `URI` 头里的URL。这样可以避免在Apache作为反向代理使用时，后端服务器的HTTP重定向造成的绕过反向代理的问题。

只有明确指定的应答头会被重写，其它应答头保持不变，并且HTML页面中的URL也不会被修改。如果被代理的内容包含绝对URL引用，那么将会绕过代理。有一个第三方模块可以检查并改写HTML中的URL引用，该模块就是Nick Kew编写的mod_proxy_html。

path是本地虚拟路径的名称。url是远端服务器的部分URL。与 `ProxyPass` 指令中的使用方法相同。

例如，假定本地服务器拥有地址 `http://example.com/` ，那么

```
ProxyPass         /mirror/foo/ http://backend.example.com/
ProxyPassReverse  /mirror/foo/ http://backend.example.com/
ProxyPassReverseCookieDomain  backend.example.com  public.example.com
ProxyPassReverseCookiePath  /  /mirror/foo/
```

不仅会把所有对 `http://example.com/mirror/foo/bar` 的请求直接转换为对 `http://backend.example.com/bar` 的代理请求(由 `ProxyPass` 提供的功能)，它还会重定向服务器 `backend.example.com` 的发送：当 `http://backend.example.com/bar` 被它重定向到 `http://backend.example.com/quux` 时，Apache会在转交HTTP重定向应答到客户端之前调整它为 `http://example.com/mirror/foo/quux` 。注意：被用于构建URL的主机名与 `UseCanonicalName` 指令的设置有关。

注意，此 `ProxyPassReverse` 指令亦可与 `mod_rewrite` 的代理穿透特性(`RewriteRule ... [P]`)联用。因为它不依赖于相应的 `ProxyPass` 指令。

当在 `<Location>` 配置段中使用时，第一个参数会被忽略而采用由 `<Location>` 指令指定的本地目录。

ProxyPassReverseCookieDomain 指令

说明	**Adjusts the Domain string in Set-Cookie headers from a reverse- proxied server**
语法	`ProxyPassReverseCookieDomain internal-domain public-domain`
作用域	server config, virtual host, directory
状态	扩展(E)
模块	mod_proxy

Usage is basically similar to `ProxyPassReverse` , but instead of rewriting headers that are a URL, this rewrites the `domain` string in `Set-Cookie` headers.

ProxyPassReverseCookiePath 指令

说明	**Adjusts the Path string in Set-Cookie headers from a reverse- proxied server**
语法	`ProxyPassReverseCookiePath internal-path public-path`
作用域	server config, virtual host, directory
状态	扩展(E)
模块	mod_proxy

Usage is basically similar to `ProxyPassReverse` , but instead of rewriting headers that are a URL, this rewrites the `path` string in `Set-Cookie` headers.

ProxyPreserveHost 指令

说明	使用进入的**HTTP**请求头来发送代理请求
语法	`ProxyPreserveHost On|Off`
默认值	`ProxyPreserveHost Off`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy
兼容性	仅在 Apache 2.0.31 及以后的版本中可用

当启用时，此选项将把传入请求的"Host:"行传递给被代理的主机，而不是传递在 `ProxyPass` 中指定的主机名。

此选项一般为 `Off` 状态。It is mostly useful in special configurations like proxied mass name-based virtual hosting, where the original Host header needs to be evaluated by the backend server.

ProxyReceiveBufferSize 指令

说明	代理HTTP和FTP连接的接收缓冲区大小(字节)
语法	`ProxyReceiveBufferSize bytes`
默认值	`ProxyReceiveBufferSize 0`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

`ProxyReceiveBufferSize` 指令为增加的吞吐量指定了代理HTTP和FTP连接的(TCP/IP)网络接收缓冲区。这个值必须大于 `512` ，或设置为" `0` "表示使用系统默认的缓冲大小。

示例

```
ProxyReceiveBufferSize 2048
```

ProxyRemote 指令

说明	用于处理某些特定请求的远端代理
语法	`ProxyRemote match remote-server`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

此指令定义了此代理的远端代理。match可以是远端服务器支持的URL形式的名称、或是远端服务器使用的部分URL、或是代表服务器可以接受所有请求的" `*` "。remote-server是远端服务器的部分URL。语法为：

```
<dfn class="calibre40">remote-server</dfn> = <var class="calibre40">scheme</var>://<var c
```

scheme是与远端服务器交换信息时使用的协议；本模块暂时只支持 `http` 协议。

示例

```
ProxyRemote http://goodguys.com/ http://mirrorguys.com:8000

ProxyRemote * http://cleversite.com

ProxyRemote ftp http://ftpproxy.mydomain.com:8080
```

在最后一个例子中，代理会将封装到另外一个HTTP代理请求中的FTP请求转交到另外一个能处理它们的代理去。

此选项也支持反向代理配置：一个后端web服务器可以被嵌入到一个虚拟主机的URL空间中，哪怕它是由另一个代理转交过来的。

ProxyRemoteMatch 指令

说明	处理匹配正则表达式的请求的远端代理
语法	`ProxyRemoteMatch regex remote-server`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

`ProxyRemoteMatch` 与 `ProxyRemote` 令基本相同。除了第一个参数是由一个请求的URL变成了匹配的正则表达式。

ProxyRequests 指令

说明	启用正向(标准)代理请求
语法	`ProxyRequests On|Off`
默认值	`ProxyRequests Off`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

此指令将允许或禁止Apache作为正向代理服务器的功能(设置为 `Off` 并不会禁用 `ProxyPass` 指令)。

在一个典型的反向代理配置中，此可选项一般设置为 `Off` 。

为了能够代理HTTP或FTP站点， `mod_proxy_http` 或 `mod_proxy_ftp` 必须同时存在于服务器中。

警告

在您没有对服务器采取安全措施之前，请不要用 `ProxyRequests` 启用您的代理。一个开放的代理服务器不仅对您的网络有威胁，对整个因特网来说也同样如此。

ProxyTimeout 指令

说明	代理请求的网络超时
语法	`ProxyTimeout seconds`
默认值	`ProxyTimeout 300`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy
兼容性	仅在 Apache 2.0.31 及以后的版本中可用

此指令允许用户对代理请求指定一个超时值。当你有一个很慢/错误多多的应用服务器经常挂起，而您宁愿返回一个超时的失败信息也不愿意继续等待不知道多久的时候，这个功能是很有用的。

ProxyVia 指令

说明	控制代理对 **`Via`** 应答头的使用
语法	`ProxyVia On|Off|Full|Block`
默认值	`ProxyVia Off`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_proxy

此指令控制代理对"`Via:`"头的使用。它的目的是控制位于代理服务器链中的代理请求的流向。参阅RFC 2616(HTTP/1.1)14.45小节以获得关于"`Via:`"头的解释。

- 如果设置为默认值 `Off` ，将不会采取特殊的处理。如果一个请求或应答包含"`Via:`"头，将不进行任何修改而直接通过。
- 如果设置为 `On` 每个请求和应答都会对应当前主机得到一个"`Via:`"头。
- 如果设置为 `Full` ，每个产生的"`Via:`"头中都会额外加入Apache服务器的版本，以"`Via:`"注释域出现。
- 如果设置为 `Block` ，每个代理请求中的所有"`Via:`"头行都将被删除。且不会产生新的"`Via:`"头。

Apache模块 mod_proxy_ajp

说明	`mod_proxy` 的扩展，提供**Apache JServ Protocol**支持
状态	扩展(E)
模块名	proxy_ajp_module
源文件	proxy_ajp.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

This module *requires* the service of `mod_proxy` . It provides support for the `Apache JServ Protocol version 1.3` (hereafter *AJP13*).

Thus, in order to get the ability of handling `AJP13` protocol, `mod_proxy` 和 `mod_proxy_ajp` have to be present in the server.

警告

在您没有对您的服务器采取安全措施之前，不要启用代理。开放的代理服务器对你自己的内部网络和大规模的Internet网都是有安全隐患的。

Overview of the protocol

`AJP13` protocol is packet-oriented. A binary format was presumably chosen over the more readable plain text for reasons of performance. The web server communicates with the servlet container over TCP connections. To cut down on the expensive process of socket creation, the web server will attempt to maintain persistent TCP connections to the servlet container, and to reuse a connection for multiple request/response cycles.

Once a connection is assigned to a particular request, it will not be used for any others until the request-handling cycle has terminated. In other words, requests are not multiplexed over connections. This makes for much simpler code at either end of the connection, although it does cause more connections to be open at once.

Once the web server has opened a connection to the servlet container, the connection can be in one of the following states:

- Idle No request is being handled over this connection.
- Assigned The connecton is handling a specific request.

Once a connection is assigned to handle a particular request, the basic request informaton (e.g. HTTP headers, etc) is sent over the connection in a highly condensed form (e.g. common strings are encoded as integers). Details of that format are below in Request Packet Structure. If there is a body to the request `(content-length > 0)` , that is sent in a separate packet immediately after.

At this point, the servlet container is presumably ready to start processing the request. As it does so, it can send the following messages back to the web server:

- SEND_HEADERS Send a set of headers back to the browser.
- SEND_BODY_CHUNK Send a chunk of body data back to the browser.
- GET_BODY_CHUNK Get further data from the request if it hasn't all been transferred yet. This is necessary because the packets have a fixed maximum size and arbitrary amounts of data can be included the body of a request (for uploaded files, for example). (Note: this is unrelated to HTTP chunked tranfer).
- END_RESPONSE Finish the request-handling cycle.

Each message is accompanied by a differently formatted packet of data. See Response Packet Structures below for details.

Basic Packet Structure

There is a bit of an XDR heritage to this protocol, but it differs in lots of ways (no 4 byte alignment, for example).

Byte order: I am not clear about the endian-ness of the individual bytes. I'm guessing the bytes are little-endian, because that's what XDR specifies, and I'm guessing that sys/socket library is magically making that so (on the C side). If anyone with a better knowledge of socket calls can step in, that would be great.

There are four data types in the protocol: bytes, booleans, integers and strings.

Byte

A single byte.

Boolean

A single byte, `1 = true` , `0 = false` . Using other non-zero values as true (i.e. C-style) may work in some places, but it won't in others.

Integer

A number in the range of `0 to 2^16 (32768)` . Stored in 2 bytes with the high-order byte first.

String

A variable-sized string (length bounded by 2^16). Encoded with the length packed into two bytes first, followed by the string (including the terminating '\0'). Note that the encoded length does **not** include the trailing '\0' -- it is like `strlen`. This is a touch confusing on the Java side, which is littered with odd autoincrement statements to skip over these terminators. I believe the reason this was done was to allow the C code to be extra efficient when reading strings which the servlet container is sending back -- with the terminating \0 character, the C code can pass around references into a single buffer, without copying. if the \0 was missing, the C code would have to copy things out in order to get its notion of a string.

Packet Size

According to much of the code, the max packet size is `8 * 1024 bytes (8K)`. The actual length of the packet is encoded in the header.

Packet Headers

Packets sent from the server to the container begin with `0x1234`. Packets sent from the container to the server begin with `AB` (that's the ASCII code for A followed by the ASCII code for B). After those first two bytes, there is an integer (encoded as above) with the length of the payload. Although this might suggest that the maximum payload could be as large as 2^16, in fact, the code sets the maximum to be 8K.

Packet Format (Server->Container)					
Byte	0	1	2	3	4...(n+3)
Contents	0x12	0x34	Data Length (n)	Data	

Packet Format (Container->Server)					
Byte	0	1	2	3	4...(n+3)
Contents	A	B	Data Length (n)	Data	

For most packets, the first byte of the payload encodes the type of message. The exception is for request body packets sent from the server to the container -- they are sent with a standard packet header (`0x1234` and then length of the packet), but without any prefix code after that.

The web server can send the following messages to the servlet container:

Code	Type of Packet	Meaning
2	Forward Request	Begin the request-processing cycle with the following data
7	Shutdown	The web server asks the container to shut itself down.
8	Ping	The web server asks the container to take control (secure login phase).
10	CPing	The web server asks the container to respond quickly with a CPong.
none	Data	Size (2 bytes) and corresponding body data.

To ensure some basic security, the container will only actually do the `Shutdown` if the request comes from the same machine on which it's hosted.

The first `Data` packet is send immediatly after the `Forward Request` by the web server.

The servlet container can send the following types of messages to the webserver:

Code	Type of Packet	Meaning
3	Send Body Chunk	Send a chunk of the body from the servlet container to the web server (and presumably, onto the browser).
4	Send Headers	Send the response headers from the servlet container to the web server (and presumably, onto the browser).
5	End Response	Marks the end of the response (and thus the request-handling cycle).
6	Get Body Chunk	Get further data from the request if it hasn't all been transferred yet.
9	CPong Reply	The reply to a CPing request

Each of the above messages has a different internal structure, detailed below.

Request Packet Structure

For messages from the server to the container of type *Forward Request*:

```
AJP13_FORWARD_REQUEST :=
    prefix_code      (byte) 0x02 = JK_AJP13_FORWARD_REQUEST
    method           (byte)
    protocol         (string)
    req_uri          (string)
    remote_addr      (string)
    remote_host      (string)
    server_name      (string)
    server_port      (integer)
    is_ssl           (boolean)
    num_headers      (integer)
    request_headers *(req_header_name req_header_value)
    attributes      *(attribut_name attribute_value)
    request_terminator (byte) OxFF
```

`request_headers` have the following structure:

```
req_header_name :=
    sc_req_header_name | (string)  [see below for how this is parsed]

sc_req_header_name := 0xA0xx (integer)

req_header_value := (string)
```

`attributes` are optional and have the following structure:

```
attribute_name := sc_a_name | (sc_a_req_attribute string)

attribute_value := (string)
```

Not that the all-important header is `content-length`, because it determines whether or not the container looks for another packet immediately.

Detailed description of the elements of Forward Request

Request prefix

For all requests, this will be 2. See above for details on other Prefix codes.

Method

The HTTP method, encoded as a single byte:

Command Name	Code
OPTIONS	1
GET	2
HEAD	3
POST	4
PUT	5
DELETE	6
TRACE	7
PROPFIND	8
PROPPATCH	9
MKCOL	10
COPY	11
MOVE	12
LOCK	13
UNLOCK	14
ACL	15
REPORT	16
VERSION-CONTROL	17
CHECKIN	18
CHECKOUT	19
UNCHECKOUT	20
SEARCH	21
MKWORKSPACE	22
UPDATE	23
LABEL	24
MERGE	25
BASELINE_CONTROL	26
MKACTIVITY	27

Later version of ajp13, will transport additional methods, even if they are not in this list.

protocol, req_uri, remote_addr, remote_host, server_name, server_port, is_ssl

These are all fairly self-explanatory. Each of these is required, and will be sent for every request.

Headers

The structure of `request_headers` is the following: First, the number of headers `num_headers` is encoded. Then, a series of header name `req_header_name` / value `req_header_value` pairs follows. Common header names are encoded as integers, to save space. If the header name is not in the list of basic headers, it is encoded normally (as a string, with prefixed length). The list of common headers `sc_req_header_name` and their codes is as follows (all are case-sensitive):

Name	Code value	Code name
accept	0xA001	SC_REQ_ACCEPT
accept-charset	0xA002	SC_REQ_ACCEPT_CHARSET
accept-encoding	0xA003	SC_REQ_ACCEPT_ENCODING
accept-language	0xA004	SC_REQ_ACCEPT_LANGUAGE
authorization	0xA005	SC_REQ_AUTHORIZATION
connection	0xA006	SC_REQ_CONNECTION
content-type	0xA007	SC_REQ_CONTENT_TYPE
content-length	0xA008	SC_REQ_CONTENT_LENGTH
cookie	0xA009	SC_REQ_COOKIE
cookie2	0xA00A	SC_REQ_COOKIE2
host	0xA00B	SC_REQ_HOST
pragma	0xA00C	SC_REQ_PRAGMA
referer	0xA00D	SC_REQ_REFERER
user-agent	0xA00E	SC_REQ_USER_AGENT

The Java code that reads this grabs the first two-byte integer and if it sees an `'0xA0'` in the most significant byte, it uses the integer in the second byte as an index into an array of header names. If the first byte is not `0xA0` , it assumes that the two-byte integer is the length of a string, which is then read in.

This works on the assumption that no header names will have length greater than `0x9999 (==0xA000 - 1)` , which is perfectly reasonable, though somewhat arbitrary.

注意：

The `content-length` header is extremely important. If it is present and non-zero, the container assumes that the request has a body (a POST request, for example), and immediately reads a separate packet off the input stream to get that body.

Attributes

The attributes prefixed with a `?` (e.g. `?context`) are all optional. For each, there is a single byte code to indicate the type of attribute, and then a string to give its value. They can be sent in any order (thogh the C code always sends them in the order listed below). A special terminating code is sent to signal the end of the list of optional attributes. The list of byte codes is:

Information	Code Value	Note
?context	0x01	Not currently implemented
?servlet_path	0x02	Not currently implemented
?remote_user	0x03	
?auth_type	0x04	
?query_string	0x05	
?jvm_route	0x06	
?ssl_cert	0x07	
?ssl_cipher	0x08	
?ssl_session	0x09	
?req_attribute	0x0A	Name (the name of the attribute follows)
?ssl_key_size	0x0B	
are_done	0xFF	request_terminator

`context` 和 `servlet_path` are not currently set by the C code, and most of the Java code completely ignores whatever is sent over for those fields (and some of it will actually break if a string is sent along after one of those codes). I don't know if this is a bug or an unimplemented feature or just vestigial code, but it's missing from both sides of the connection.

`remote_user` 和 `auth_type` presumably refer to HTTP-level authentication, and communicate the remote user's username and the type of authentication used to establish their identity (e.g. Basic, Digest).

`query_string` , `ssl_cert` , `ssl_cipher` , and `ssl_session` refer to the corresponding pieces of HTTP and HTTPS.

`jvm_route` , is used to support sticky sessions -- associating a user's sesson with a particular Tomcat instance in the presence of multiple, load-balancing servers.

Beyond this list of basic attributes, any number of other attributes can be sent via the `req_attribute` code `0x0A` . A pair of strings to represent the attribute name and value are sent immediately after each instance of that code. Environment values are passed in via this method.

Finally, after all the attributes have been sent, the attribute terminator, `0xFF` , is sent. This signals both the end of the list of attributes and also then end of the Request Packet.

Response Packet Structure

for messages which the container can send back to the server.

```
AJP13_SEND_BODY_CHUNK :=
  prefix_code   3
  chunk_length  (integer)
  chunk        *(byte)

AJP13_SEND_HEADERS :=
  prefix_code       4
  http_status_code  (integer)
  http_status_msg   (string)
  num_headers       (integer)
  response_headers *(res_header_name header_value)

res_header_name :=
    sc_res_header_name | (string)   [see below for how this is parsed]

sc_res_header_name := 0xA0 (byte)

header_value := (string)

AJP13_END_RESPONSE :=
  prefix_code       5
  reuse             (boolean)

AJP13_GET_BODY_CHUNK :=
  prefix_code       6
  requested_length  (integer)
```

Details:

Send Body Chunk

The chunk is basically binary data, and is sent directly back to the browser.

Send Headers

The status code and message are the usual HTTP things (e.g. `200` 和 `OK`). The response header names are encoded the same way the request header names are. See header_encoding above for details about how the the codes are distinguished from the strings. The codes for common headers are:

Name	Code value
Content-Type	0xA001
Content-Language	0xA002
Content-Length	0xA003
Date	0xA004
Last-Modified	0xA005
Location	0xA006
Set-Cookie	0xA007
Set-Cookie2	0xA008
Servlet-Engine	0xA009
Status	0xA00A
WWW-Authenticate	0xA00B

After the code or the string header name, the header value is immediately encoded.

End Response

Signals the end of this request-handling cycle. If the `reuse` flag is true `(==1)`, this TCP connection can now be used to handle new incoming requests. If `reuse` is false (anything other than 1 in the actual C code), the connection should be closed.

Get Body Chunk

The container asks for more data from the request (If the body was too large to fit in the first packet sent over or when the request is chuncked). The server will send a body packet back with an amount of data which is the minimum of the `request_length`, the maximum send body size `(8186 (8 Kbytes - 6))`, and the number of bytes actually left to send from the request body. If there is no more data in the body (i.e. the servlet container is trying to read past the end of the body), the server will send back an *empty* packet, which is a body packet with a payload length of 0. `(0x12,0x34,0x00,0x00)`

Apache模块 mod_proxy_balancer

说明	mod_proxy 的扩展，提供负载平衡支持
状态	扩展(E)
模块名	proxy_balancer_module
源文件	proxy_balancer.c
兼容性	仅在 Apache 2.1 及以后的版本中可用

概述

This module *requires* the service of `mod_proxy` . It provides load balancing support for `HTTP` , `FTP` 和 `AJP13` protocols

Thus, in order to get the ability of load balancing, `mod_proxy` 和 `mod_proxy_balancer` have to be present in the server.

警告

在您没有对您的服务器采取安全措施之前，不要启用代理。开放的代理服务器对你自己的内部网络和大规模的Internet网都是有安全隐患的。

Load balancer scheduler algorithm

At present, there are 2 load balancer scheduler algorithms available for use: Request Counting and Weighted Traffic Counting. These are controlled via the `lbmethod` value of the Balancer definition. See the `Proxy` directive for more information.

Request Counting Algorithm

Enabled via `lbmethod=byrequests` , the idea behind this scheduler is that we distribute the requests among the various workers to ensure that each gets their configured share of the number of requests. It works as follows:

<dfn class="calibre27">lbfactor</dfn> is *how much we expect this worker to work*, or *the workers's work quota*. This is a normalized value representing their "share" of the amount of work to be done.

<dfn class="calibre27">lbstatus</dfn> is *how urgent this worker has to work to fulfill its quota of work*.

<dfn class="calibre27">worker</dfn> is a member of the load balancer, usually a remote host serving one of the supported protocols.

We distribute each worker's work quota to the worker, and then look which of them needs to work most urgently (biggest lbstatus). This worker is then selected for work, and its lbstatus reduced by the total work quota we distributed to all workers. Thus the sum of all lbstatus does not change(*) and we distribute the requests as desired.

If some workers are disabled, the others will still be scheduled correctly.

```
for each worker in workers
    worker lbstatus += worker lbfactor
    total factor    += worker lbfactor
    if worker lbstatus > candidate lbstatus
        candidate = worker

candidate lbstatus -= total factor
```

If a balancer is configured as follows:

worker	a	b	c	d
lbfactor	25	25	25	25
lbstatus	0	0	0	0
---	---	---		

And b gets disabled, the following schedule is produced:

worker	a	b	c	d
lbstatus	*-50*	0	25	25
lbstatus	-25	0	*-25*	50
lbstatus	0	0	0	*0*
(repeat)				

That is it schedules: a c d a c d a c d ... Please note that:

worker	a	b	c	d
lbfactor	25	25	25	25
---	---	---		

Has the exact same behavior as:

worker	a	b	c	d
lbfactor	1	1	1	1
---	---	---		

This is because all values of <dfn class="calibre27">lbfactor</dfn> are normalized with respect to the others. For:

worker	a	b	c
lbfactor	1	4	1
---	---		

worker b will, on average, get 4 times the requests that a和c will.

The following asymmetric configuration works as one would expect:

worker	a	b
lbfactor	70	30
lbstatus	*-30*	30
lbstatus	40	*-40*
lbstatus	*10*	-10
lbstatus	*-20*	20
lbstatus	*-50*	50
lbstatus	20	*-20*
lbstatus	*-10*	10
lbstatus	*-40*	40
lbstatus	30	*-30*
lbstatus	*0*	0
(repeat)		

That is after 10 schedules, the schedule repeats and 7 a are selected with 3 b interspersed.

Weighted Traffic Counting Algorithm

Enabled via `lbmethod=bytraffic` , the idea behind this scheduler is very similar to the Request Counting method, with the following changes:

<dfn class="calibre27">lbfactor</dfn> is *how much traffic, in bytes, we want this worker to handle*. This is also a normalized value representing their "share" of the amount of work to be done, but instead of simply counting the number of requests, we take into account the amount of traffic this worker has seen.

If a balancer is configured as follows:

worker	a	b	c
lbfactor	1	2	1
---	---		

Then we mean that we want b to process twice the amount of bytes than a或c should. It does not necessarily mean that b would handle twice as many requests, but it would process twice the I/O. Thus, the size of the request and response are applied to the weighting and selection algorithm.

Enabling Balancer Manager Support

This module *requires* the service of `mod_status`. Balancer manager enables dynamic update of balancer members. You can use balancer manager to change the balance factor or a particular member, or put it in the off line mode.

Thus, in order to get the ability of load balancer management, `mod_status`和`mod_proxy_balancer` have to be present in the server.

To enable load balancer management for browsers from the foo.com domain add this code to your `httpd.conf` configuration file

```
<Location /balancer-manager>

SetHandler balancer-manager

Order Deny,Allow

Deny from all

Allow from .foo.com

</Location>
```

You can now access load balancer manager by using a Web browser to access the page `http://your.server.name/balancer-manager`

Apache模块 mod_proxy_connect

说明	mod_proxy 的扩展，提供对处理HTTP CONNECT 方法的支持
状态	扩展(E)
模块名	proxy_connect_module
源文件	proxy_connect.c

概述

本模块需要 mod_proxy 提供的服务。它提供对HTTP的 CONNECT 方法的支持。这个方法主要用于处理通过代理服务器的隧道SSL请求。

为了能处理 CONNECT 请求，模块 mod_proxy 和 mod_proxy_connect 必须同时存在于服务器中。

警告

在您没有对您的服务器采取安全措施之前，不要启用代理。开放的代理服务器对你自己的内部网络和大规模的Internet网都是有安全隐患的。

Apache模块 mod_proxy_ftp

说明	`mod_proxy` 的FTP支持模块
状态	扩展(E)
模块名	proxy_ftp_module
源文件	proxy_ftp.c

概述

本模块提供了代理FTP站点的能力，它需要 `mod_proxy` 提供的服务。这样，为了能处理FTP代理请求，模块 `mod_proxy` 和 `mod_proxy_ftp` 必须同时存在于服务器中。

注意：目前对FTP的支持仅限于GET方法。

警告

在您没有对您的服务器采取安全措施之前，不要启用代理。开放的代理服务器对你自己的内部网络和大规模的Internet网都是有安全隐患的。

为什么xxx类型的文件不能从FTP下载？

您可能没有在您的代理mime类型配置文件中定义特定的文件类型 `application/octet-stream`。有用的一行可能是这样的：

```
application/octet-stream   bin dms lha lzh exe class tgz taz
```

作为另一种选择，你可能希望将所有东西都看成二进制类型：

```
DefaultType application/octet-stream
```

如何强制文件xxx使用FTP的ASCII形式下载？

在很罕见的情况下，也许您想要用FTP的 `ASCII` 传输模式(默认是 `binary` 模式)来下载某个特定的文件，您可以用在请求前面加上" `;type=a` "前缀的方式覆盖 `mod_proxy` 的默认值来强制进行ASCII模式的传输。但是不论如何，FTP目录列表将始终以ASCII模式执行。

我如何使用FTP上传？

目前，mod_proxy仅支持FTP的GET方法，因此不支持使用FTP上传。你可以通过Apache代理改用HTTP上传(POST或PUT)。

我如何能访问我自己home目录以外的FTP文件？

一个FTP URI一般被当成登录用户home目录的相对路径处理。唉，可惜您不能使用"/../"来到达更上层的目录，因为点(.)由浏览器解释而不会真正发送给FTP服务器。为搞定这个问题，在Apache FTP代理中实现了一个"<dfn class="calibre27">Squid %2f hack</dfn>"。这是一个也被其它流行的类似Squid Proxy Cache的代理服务器使用的解决方法。使用预先将" `/%2f` "加入您请求路径的方法，您能使代理将FTP起始目录改为" `/` "(而不是home目录)。例如，为了取得文件 `/etc/motd` ，您应当使用下面这样的URL ：

```
ftp://<var class="calibre40">user</var>@<var class="calibre40">host</var>/%2f/etc
```

我如何才能在浏览器的URL框中隐藏FTP的明文密码？

使用用户名和密码登入一个FTP服务器时，Apache使用了不同的策略。当URL中不存在用户名和密码时，Apache会向FTP服务器发出一个匿名用户的登录，比如说：

```
user: anonymous
password: apache_proxy@
```

这对于配置了匿名访问的大多数FTP服务器来说是很有效的。

要使用特定的用户名，可以将这个特定的用户名嵌入URL中：

```
ftp://<var class="calibre40">username</var>@<var class="calibre40">host</var>/myf
```

如果在给出了这个用户名后，FTP服务器要求提供一个密码(这是它应该做的)，这时Apache会回应一个" `401` "(需要认证)应答，这将会使浏览器弹出一个用户名/密码对话框。当输入了密码后，将会再次尝试连接，如果成功，则请求的资源就会被下载。这种方法的好处在于您的浏览器不会以明码的形式显示密码，而当您使用

```
ftp://<var class="calibre40">username</var>:<var class="calibre40">password</var>
```

的时候就无法做到这一点。

注意

这种方法提交的密码在传输的时候没有进行加密。它在您的浏览器到Apache代理服务器之间传输时为base64格式的明文字符串，而在Apache代理服务器和FTP服务器之间传输的为普通文本。所以，在通过HTTP访问您的FTP服务器之前(或通过FTP访问您的私人文件之前)您应该慎重考虑一下。当使用这种不安全的手段时，一个窃听者可能会用这种方法截取您的密码。

Apache模块 mod_proxy_http

说明	`mod_proxy` 的HTTP支持模块
状态	扩展(E)
模块名	proxy_http_module
源文件	proxy_http.c

概述

本模块需要 `mod_proxy` 提供的服务。它提供代理HTTP请求的功能。 `mod_proxy_http` 支持HTTP/0.9, HTTP/1.0, HTTP/1.1 标准。它不提供任何缓冲能力。如果你想要设置使用缓存的代理，可以使用 `mod_cache` 模块提供的服务。

这样，为了能处理HTTP代理请求，模块 `mod_proxy` 和 `mod_proxy_http` 必须同时存在于服务器中。

警告

在您没有对您的服务器采取安全措施之前，不要启用代理。开放的代理服务器对你自己的内部网络和大规模的Internet网都是有安全隐患的。

Apache模块 mod_rewrite

说明	一个基于一定规则的实时重写URL请求的引擎
状态	扩展(E)
模块名	rewrite_module
源文件	mod_rewrite.c
兼容性	仅在 Apache 1.3 及以后的版本中可用

概述

此模块提供了一个基于正则表达式分析器的重写引擎来实时重写URL请求。它支持每个完整规则可以拥有不限数量的子规则以及附加条件规则的灵活而且强大的URL操作机制。此URL操作可以依赖于各种测试，比如服务器变量、环境变量、HTTP头、时间标记，甚至各种格式的用于匹配URL组成部分的查找数据库。

此模块可以操作URL的所有部分(包括路径信息部分)，在服务器级的(`httpd.conf`)和目录级的(`.htaccess`)配置都有效，还可以生成最终请求字符串。此重写操作的结果可以是内部子处理，也可以是外部请求的转向，甚至还可以是内部代理处理。

但是，所有这些功能和灵活性带来一个问题，那就是复杂性，因此，不要指望一天之内就能看懂整个模块。

更多的讨论、细节、示例，请查看详细的URL重写文档。

特殊字符的引用

在Apache 1.3.20中，*TestString*和*Substitution*中的特殊字符可以用前导斜杠()来实现转义(即忽略其特殊含义而视之为普通字符)。 比如，*Substitution*可以用" `\$` "来包含一个美元符号，以避免mod_rewrite把它视为反向引用。

环境变量

此模块会跟踪两个额外的(非标准)CGI/SSI环境变量， `SCRIPT_URL` 和 `SCRIPT_URI` 。他们包含了当前资源的逻辑网络视图，而标准CGI/SSI变量 `SCRIPT_NAME` 和 `SCRIPT_FILENAME` 包含的是物理系统视图。

注意：这些变量保持的是其最初被请求时的_*URI/URL*，即在任何重写操作之前_的URI/URL。其重要性在于他们是重写操作重写URL到物理路径名的原始依据。

示例

```
SCRIPT_NAME=/sw/lib/w3s/tree/global/u/rse/.www/index.html
SCRIPT_FILENAME=/u/rse/.www/index.html
SCRIPT_URL=/u/rse/
SCRIPT_URI=http://en1.engelschall.com/u/rse/
```

实用方案

我们提供了URL重写指南和高级URL重写指南文档，列举了许多基于URL的问题的实用方案，其中你可以找到真实有用的规则集。

RewriteBase 指令

说明	设置目录级重写的基准URL
语法	`RewriteBase _URL-path_`
默认值	参见使用方法
作用域	directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_rewrite

`RewriteBase` 指令显式地设置了目录级重写的基准URL。在下文中，你可以看见 `RewriteRule` 可以用于目录级的配置文件中(`.htaccess`)并在局部范围内起作用，即规则实际处理的只是剥离了本地路径前缀的一部分。处理结束后，这个路径会被自动地附着回去。默认值是" `RewriteBase` *physical-directory-path*"。

在对一个新的URL进行替换时，此模块必须把这个URL重新注入到服务器处理中。为此，它必须知道其对应的URL前缀或者说URL基准。通常，此前缀就是对应的文件路径。但是，大多数网站**URL**不是直接对应于其物理文件路径的，因而一般不能做这样的假定! 所以在这种情况下，就必须用 `RewriteBase` 指令来指定正确的URL前缀。

如果你的网站服务器URL不是与物理文件路径直接对应的，而又需要使用 `RewriteBase` 指令，则必须在每个对应的 `.htaccess` 文件中指定 `RewriteRule` 。

例如，目录级配置文件内容如下：

```
#
#  /abc/def/.htaccess -- per-dir config file for directory /abc/def
#  Remember: /abc/def is the physical path of /xyz, _i.e._, the server
#            has a 'Alias /xyz /abc/def' directive 例如,
#

RewriteEngine On

#  let the server know that we were reached via /xyz and not
#  via the physical path prefix /abc/def
RewriteBase   /xyz

#  now the rewriting rules
RewriteRule   ^oldstuff\.html$  newstuff.html
```

上述例子中，对 `/xyz/oldstuff.html` 的请求被正确地重写为物理的文件 `/abc/def/newstuff.html` 。

For Apache Hackers

以下列出了内部处理的详细步骤：

```
Request:
  /xyz/oldstuff.html

Internal Processing:
  /xyz/oldstuff.html     -> /abc/def/oldstuff.html  (per-server Alias)
  /abc/def/oldstuff.html -> /abc/def/newstuff.html  (per-dir    RewriteRule)
  /abc/def/newstuff.html -> /xyz/newstuff.html      (per-dir    RewriteBase)
  /xyz/newstuff.html     -> /abc/def/newstuff.html  (per-server Alias)

Result:
  /abc/def/newstuff.html
```

虽然这个过程看来很繁复，但是由于目录级重写的到来时机已经太晚了，它不得不把这个(重写)请求重新注入到Apache核心中，所以Apache内部确实是这样处理的。但是：它的开销并不象看起来的那样大，因为重新注入完全在Apache服务器内部进行，而且这样的过程在Apache内部也为其他许多操作所使用。所以，你可以充分信任其设计和实现是正确的。

RewriteCond 指令

说明	定义重写发生的条件
语法	`RewriteCond _TestString_ _CondPattern_`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_rewrite

`RewriteCond` 指令定义了一个规则的条件，即在一个 `RewriteRule` 指令之前有一个或多个 `RewriteCond` 指令。条件之后的重写规则仅在当前URI与pattern匹配并且符合这些条件的时候才会起作用。

*TestString*是一个纯文本的字符串，但是还可以包含下列可扩展的成分：

- **RewriteRule**反向引用，引用方法是：

  `$N`

  (0 <= N <= 9)引用当前(带有若干 `RewriteRule` 指令的) `RewriteCond` 中的与pattern匹配的分组成分(圆括号!)。

- **RewriteCond**反向引用，引用方法是：

  `%N`

  (1 <= N <= 9)引用当前若干 `RewriteCond` 条件中最后符合的条件中的分组成分(圆括号!)。

- **RewriteMap**扩展，引用方法是：

  `${mapname:key|default}`

  细节请参见RewriteMap 指令。

- 服务器变量，引用方法是：

  `%{` ***NAME_OF_VARIABLE*** `}`

  *NAME_OF_VARIABLE*可以是下表列出的字符串之一：

HTTP headers:	connection & request:
HTTP_USER_AGENT HTTP_REFERER HTTP_COOKIE HTTP_FORWARDED HTTP_HOST HTTP_PROXY_CONNECTION HTTP_ACCEPT	REMOTE_ADDR REMOTE_HOST REMOTE_PORT REMOTE_USER REMOTE_IDENT REQUEST_METHOD SCRIPT_FILENAME PATH_INFO QUERY_STRING AUTH_TYPE

server internals:	date and time:	specials:
DOCUMENT_ROOT SERVER_ADMIN SERVER_NAME SERVER_ADDR SERVER_PORT SERVER_PROTOCOL SERVER_SOFTWARE	TIME_YEAR TIME_MON TIME_DAYTIME_HOUR TIME_MINTIME_SEC TIME_WDAY TIME	API_VERSION THE_REQUEST REQUEST_URI REQUEST_FILENAME IS_SUBREQ HTTPS

These variables all correspond to the similarly named HTTP MIME-headers, C variables of the Apache server or `struct tm` fields of the Unix system. Most are documented elsewhere in the Manual or in the CGI specification. Those that are special to mod_rewrite include:

`IS_SUBREQ`

Will contain the text "true" if the request currently being processed is a sub-request, "false" otherwise. Sub-requests may be generated by modules that need to resolve additional files or URIs in order to complete their tasks.

`API_VERSION`

This is the version of the Apache module API (the internal interface between server and module) in the current httpd build, as defined in include/ap_mmn.h. The module API version corresponds to the version of Apache in use (in the release version of Apache 1.3.14, for instance, it is 19990320:10), but is mainly of interest to module authors.

`THE_REQUEST`

The full HTTP request line sent by the browser to the server (e.g., "`GET /index.html HTTP/1.1`"). This does not include any additional headers sent by the browser.

`REQUEST_URI`

The resource requested in the HTTP request line. (In the example above, this would be "/index.html".)

`REQUEST_FILENAME`

The full local filesystem path to the file or script matching the request.

`HTTPS`

Will contain the text "on" if the connection is using SSL/TLS, or "off" otherwise. (This variable can be safely used regardless of whether `mod_ssl` is loaded).

Special Notes:

1. The variables SCRIPT*FILENAME and REQUEST_FILENAME contain the same value, _i.e.*, the value of the `filename` field of the internal `request_rec` structure of the Apache server. The first name is just the commonly known CGI variable name while the second is the consistent counterpart to REQUEST_URI (which contains the value of the `uri` field of `request_rec`).
2. There is the special format: `%{ENV:variable}` where *variable* can be any environment variable. This is looked-up via internal Apache structures and (if not found there) via `getenv()` from the Apache server process.

3. There is the special format: `%{SSL:variable}` where *variable* is the name of an SSL environment variable; this can be used whether or not `mod_ssl` is loaded, but will always expand to the empty string if it is not. Example: `%{SSL:SSL_CIPHER_USEKEYSIZE}` may expand to `128`.
4. There is the special format: `%{HTTP:header}` where *header* can be any HTTP MIME-header name. This is looked-up from the HTTP request. Example: `%{HTTP:Proxy-Connection}` is the value of the HTTP header "`Proxy-Connection:`".
5. There is the special format `%{LA-U:variable}` for look-aheads which perform an internal (URL-based) sub-request to determine the final value of *variable*. Use this when you want to use a variable for rewriting which is actually set later in an API phase and thus is not available at the current stage. For instance when you want to rewrite according to the `REMOTE_USER` variable from within the per-server context (`httpd.conf` file) you have to use `%{LA-U:REMOTE_USER}` because this variable is set by the authorization phases which come *after* the URL translation phase where mod*rewrite operates. On the other hand, because mod_rewrite implements its per-directory context (`.htaccess` file) via the Fixup phase of the API and because the authorization phases come _before* this phase, you just can use `%{REMOTE_USER}` there.
6. There is the special format: `%{LA-F:variable}` which performs an internal (filename-based) sub-request to determine the final value of *variable*. Most of the time this is the same as LA-U above.

CondPattern is the condition pattern, *i.e.*, a regular expression which is applied to the current instance of the *TestString*, *i.e.*, *TestString* is evaluated and then matched against *CondPattern*.

Remember: *CondPattern* is a *perl compatible regular expression* with some additions:

1. You can prefix the pattern string with a '`!`' character (exclamation mark) to specify a **non**-matching pattern.
2. There are some special variants of *CondPatterns*. Instead of real regular expression strings you can also use one of the following:

    - '**<CondPattern**' (is lexically lower) Treats the *CondPattern* as a plain string and compares it lexically to *TestString*. True if *TestString* is lexically lower than *CondPattern*.
    - '**>CondPattern**' (is lexically greater) Treats the *CondPattern* as a plain string and compares it lexically to *TestString*. True if *TestString* is lexically greater than *CondPattern*.
    - '**=CondPattern**' (is lexically equal) Treats the *CondPattern* as a plain string and compares it lexically to *TestString*. True if *TestString* is lexically equal to *CondPattern*, i.e the two strings are exactly equal (character by character). If *CondPattern* is just `""` (two quotation marks) this compares *TestString* to the

empty string.
  - '**-d**' (is **d**irectory) Treats the *TestString* as a pathname and tests if it exists and is a directory.
  - '**-f**' (is regular **f**ile) Treats the *TestString* as a pathname and tests if it exists and is a regular file.
  - '**-s**' (is regular file with **s**ize) Treats the *TestString* as a pathname and tests if it exists and is a regular file with size greater than zero.
  - '**-l**' (is symbolic **l**ink) Treats the *TestString* as a pathname and tests if it exists and is a symbolic link.
  - '**-x**' (has e**x**ecutable permissions) Treats the *TestString* as a pathname and tests if it exists and has execution permissions. These permissions are determined depending on the underlying OS.
  - '**-F**' (is existing file via subrequest) Checks if *TestString* is a valid file and accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to determine the check, so use it with care because it decreases your servers performance!
  - '**-U**' (is existing URL via subrequest) Checks if *TestString* is a valid URL and accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to determine the check, so use it with care because it decreases your server's performance!

Notice

All of these tests can also be prefixed by an exclamation mark ('!') to negate their meaning.

Additionally you can set special flags for *CondPattern* by appending

`[` ***flags*** `]`

as the third argument to the `RewriteCond` directive. *Flags* is a comma-separated list of the following flags:

- '**`nocase|NC`**' (**n**o **c**ase) This makes the test case-insensitive, *i.e.*, there is no difference between 'A-Z' and 'a-z' both in the expanded *TestString* and the *CondPattern*. This flag is effective only for comparisons between *TestString*和*CondPattern*. It has no effect on filesystem and subrequest checks.
- '**`ornext|OR`**' (或 next condition) Use this to combine rule conditions with a local OR instead of the implicit AND. Typical example:

```
RewriteCond %{REMOTE_HOST}  ^host1.*  [OR]
RewriteCond %{REMOTE_HOST}  ^host2.*  [OR]
RewriteCond %{REMOTE_HOST}  ^host3.*
RewriteRule ...some special stuff for any of these hosts...
```

Without this flag you would have to write the cond/rule three times.

Example:

To rewrite the Homepage of a site according to the " `User-Agent:` " header of the request, you can use the following:

```
RewriteCond  %{HTTP_USER_AGENT}  ^Mozilla.*
RewriteRule  ^/$                 /homepage.max.html  [L]

RewriteCond  %{HTTP_USER_AGENT}  ^Lynx.*
RewriteRule  ^/$                 /homepage.min.html  [L]

RewriteRule  ^/$                 /homepage.std.html  [L]
```

Interpretation: If you use Netscape Navigator as your browser (which identifies itself as 'Mozilla'), then you get the max homepage, which includes Frames, *etc.* If you use the Lynx browser (which is Terminal-based), then you get the min homepage, which contains no images, no tables, *etc.* If you use any other browser you get the standard homepage.

RewriteEngine 指令

说明	**Enables or disables runtime rewriting engine**
语法	`RewriteEngine on|off`
默认值	`RewriteEngine off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_rewrite

`RewriteEngine` directive enables or disables the runtime rewriting engine. If it is set to `off` this module does no runtime processing at all. It does not even update the `SCRIPT_URx` environment variables.

Use this directive to disable the module instead of commenting out all the `RewriteRule` directives!

Note that, by default, rewrite configurations are not inherited. This means that you need to have a `RewriteEngine on` directive for each virtual host in which you wish to use it.

RewriteLock 指令

说明	**Sets the name of the lock file used for `RewriteMap` synchronization**
语法	`RewriteLock _file-path_`
作用域	server config
状态	扩展(E)
模块	mod_rewrite

This directive sets the filename for a synchronization lockfile which mod*rewrite needs to communicate with* `RewriteMap` _*programs*. Set this lockfile to a local path (not on a NFS-mounted device) when you want to use a rewriting map-program. It is not required for other types of rewriting maps.

RewriteLog 指令

说明	**Sets the name of the file used for logging rewrite engine processing**
语法	`RewriteLog _file-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_rewrite

`RewriteLog` directive sets the name of the file to which the server logs any rewriting actions it performs. If the name does not begin with a slash (' `/` ') then it is assumed to be relative to the *Server Root*. The directive should occur only once per server config.

To disable the logging of rewriting actions it is not recommended to set *Filename* to `/dev/null`, because although the rewriting engine does not then output to a logfile it still creates the logfile output internally. **This will slow down the server with no advantage to the administrator!** To disable logging either remove or comment out the `RewriteLog` directive or use `RewriteLogLevel 0` !

安全

See the Apache Security Tips document for details on why your security could be compromised if the directory where logfiles are stored is writable by anyone other than the user that starts the server.

示例

```
RewriteLog "/usr/local/var/apache/logs/rewrite.log"
```

RewriteLogLevel 指令

说明	Sets the verbosity of the log file used by the rewrite engine
语法	`RewriteLogLevel _Level_`
默认值	`RewriteLogLevel 0`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_rewrite

`RewriteLogLevel` directive sets the verbosity level of the rewriting logfile. The default level 0 means no logging, while 9 or more means that practically all actions are logged.

To disable the logging of rewriting actions simply set *Level* to 0. This disables all rewrite action logs.

Using a high value for *Level* will slow down your Apache server dramatically! Use the rewriting logfile at a *Level* greater than 2 only for debugging!

示例

```
RewriteLogLevel 3
```

RewriteMap 指令

说明	Defines a mapping function for key-lookup
语法	`RewriteMap _MapName_ _MapType_:_MapSource_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_rewrite
兼容性	The choice of different dbm types is available in Apache 2.0.41 及以后的版本中可用

`RewriteMap` directive defines a *Rewriting Map* which can be used inside rule substitution strings by the mapping-functions to insert/substitute fields through a key lookup. The source of this lookup can be of various types.

MapName is the name of the map and will be used to specify a mapping-function for the substitution strings of a rewriting rule via one of the following constructs:

`${` ***MapName*** `:` ***LookupKey*** `}` `${` ***MapName*** `:` ***LookupKey*** `|` ***DefaultValue*** `}`

When such a construct occurs the map *MapName* is consulted and the key *LookupKey* is looked-up. If the key is found, the map-function construct is substituted by *SubstValue*. If the key is not found then it is substituted by *DefaultValue* or by the empty string if no *DefaultValue* was specified.

For example, you might define a `RewriteMap` as:

```
RewriteMap examplemap txt:/path/to/file/map.txt
```

You would then be able to use this map in a `RewriteRule` as follows:

```
RewriteRule ^/ex/(.*) ${examplemap:$1}
```

The following combinations for *MapType*和*MapSource* can be used:

- **Standard Plain Text** MapType: `txt`, MapSource: Unix filesystem path to valid regular file

  This is the standard rewriting map feature where the *MapSource* is a plain ASCII file containing either blank lines, comment lines (starting with a '#' character) or pairs like the following - one per line.

  MatchingKey SubstValue

  ## 示例

  ```
  ##
  ##  map.txt -- rewriting map
  ##

  Ralf.S.Engelschall    rse   # Bastard Operator From Hell
  Mr.Joe.Average        joe   # Mr. Average
  ```

  ```
  RewriteMap real-to-user txt:/path/to/file/map.txt
  ```

- **Randomized Plain Text** MapType: `rnd`, MapSource: Unix filesystem path to valid regular file

  This is identical to the Standard Plain Text variant above but with a special post-processing feature: After looking up a value it is parsed according to contained "`|`" characters which have the meaning of "or". In other words they indicate a set of alternatives from which the actual returned value is chosen randomly. For example, you might use the following map file and directives to provide a random load balancing between several back-end server, via a reverse-proxy. Images are sent to one of the servers in the 'static' pool, while everything else is sent to one of the 'dynamic' pool.

  Example:

  ## Rewrite map file

  ```
  ##
  ##  map.txt -- rewriting map
  ##

  static   www1|www2|www3|www4
  dynamic  www5|www6
  ```

  ## Configuration directives

  ```
  RewriteMap servers rnd:/path/to/file/map.txt

  RewriteRule ^/(.*\.(png|gif|jpg)) http://${servers:static}/$1
  [NC,P,L]

  RewriteRule ^/(.*) http://${servers:dynamic}/$1 [P,L]
  ```

- **Hash File** MapType: `dbm[=_type_]`, MapSource: Unix filesystem path to valid regular file

  Here the source is a binary format DBM file containing the same contents as a *Plain Text* format file, but in a special representation which is optimized for really fast lookups. The *type* can be sdbm, gdbm, ndbm, or db depending on compile-time settings. If the *type* is omitted, the compile-time default will be chosen. You can create such a file with any DBM tool or with the following Perl script. Be sure to adjust it to create the appropriate type of DBM. The example creates an NDBM file.

```
#!/path/to/bin/perl
##
##  txt2dbm -- convert txt map to dbm format
##

use NDBM_File;
use Fcntl;

($txtmap, $dbmmap) = @ARGV;

open(TXT, "&lt;$txtmap") or die "Couldn't open $txtmap!\n";
tie (%DB, 'NDBM_File', $dbmmap,O_RDWR|O_TRUNC|O_CREAT, 0644)
  or die "Couldn't create $dbmmap!\n";

while (&lt;TXT&gt;) {
  next if (/^\s*#/ or /^\s*$/);
  $DB{$1} = $2 if (/^\s*(\S+)\s+(\S+)/);
}

untie %DB;
close(TXT);
```

```
$ txt2dbm map.txt map.db
```

- **Internal Function** MapType: `int` , MapSource: Internal Apache function

  Here the source is an internal Apache function. Currently you cannot create your own, but the following functions already exists:

  - **toupper**: Converts the looked up key to all upper case.
  - **tolower**: Converts the looked up key to all lower case.
  - **escape**: Translates special characters in the looked up key to hex-encodings.
  - **unescape**: Translates hex-encodings in the looked up key back to special characters.
- **External Rewriting Program** MapType: `prg` , MapSource: Unix filesystem path to valid regular file

  Here the source is a program, not a map file. To create it you can use the language of your choice, but the result has to be a executable (*i.e.*, either object-code or a script with the magic cookie trick ' `#!/path/to/interpreter` ' as the first line).

  This program is started once at startup of the Apache servers and then communicates with the rewriting engine over its `stdin` 和 `stdout` file-handles. For each map-function lookup it will receive the key to lookup as a newline-terminated string on `stdin` . It then has to give back the looked-up value as a newline-terminated string on `stdout` or the four-character string " `NULL` " if it fails (*i.e.*, there is no corresponding value for the given key). A trivial program which will implement a 1:1 map (*i.e.*, key == value) could be:

```
#!/usr/bin/perl
$| = 1;
while (&lt;STDIN&gt;) {
    # ...put here any transformations or lookups...
    print $_;
}
```

But be very careful:

1. "*Keep it simple, stupid*" (KISS), because if this program hangs it will hang the Apache server when the rule occurs.
2. Avoid one common mistake: never do buffered I/O on `stdout` ! This will cause a deadloop! Hence the " `$|=1` " in the above example...
3. Use the `RewriteLock` directive to define a lockfile mod_rewrite can use to synchronize the communication to the program. By default no such synchronization takes place.

`RewriteMap` directive can occur more than once. For each mapping-function use one `RewriteMap` directive to declare its rewriting mapfile. While you cannot **declare** a map in per-directory context it is of course possible to **use** this map in per-directory context.

注意

For plain text and DBM format files the looked-up keys are cached in-core until the `mtime` of the mapfile changes or the server does a restart. This way you can have map-functions in rules which are used for **every** request. This is no problem, because the external lookup only happens once!

RewriteOptions 指令

说明	Sets some special options for the rewrite engine
语法	`RewriteOptions Options`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_rewrite
兼容性	`MaxRedirects` is no longer available in version 2.1 及以后的版本中可用

`RewriteOptions` directive sets some special options for the current per-server or per-directory configuration. The *Option* string can be currently only one:

`inherit`

This forces the current configuration to inherit the configuration of the parent. In per-virtual-server context this means that the maps, conditions and rules of the main server are inherited. In per-directory context this means that conditions and rules of the parent directory's `.htaccess` configuration are inherited.

RewriteRule 指令

说明	Defines rules for the rewriting engine
语法	`RewriteRule _Pattern_ _Substitution_`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_rewrite
兼容性	The cookie-flag is available in Apache 2.0.40 及以后的版本中可用

`RewriteRule` directive is the real rewriting workhorse. The directive can occur more than once. Each directive then defines one single rewriting rule. The **definition order** of these rules is **important**, because this order is used when applying the rules at run-time.

Pattern is a perl compatible regular expression which gets applied to the current URL. Here "current" means the value of the URL when this rule gets applied. This may not be the originally requested URL, because any number of rules may already have matched and made alterations to it.

Some hints about the syntax of regular expressions:

```
**Text:**
  **`.`**           Any single character
  **`[`**chars**`]`**     Character class: One  of chars
  **`[^`**chars**`]`**    Character class: None of chars
  text1**`|`**text2 Alternative: text1 or text2

**Quantifiers:**
  **`?`**           0 or 1 of the preceding text
  **`*`**           0 or N of the preceding text (N > 0)
  **`+`**           1 or N of the preceding text (N > 1)

**Grouping:**
  **`(`**text**`)`**      Grouping of text
            (either to set the borders of an alternative or
            for making backreferences where the **N**th group can
            be used on the RHS of a RewriteRule with $**N**)

**Anchors:**
  **`^`**           Start of line anchor
  **`$`**           End   of line anchor

**Escaping:**
  **`\`**char       escape that particular char
            (for instance to specify the chars ".[]()" _etc._)
```

For more information about regular expressions have a look at the perl regular expression manpage ("perldoc perlre"). If you are interested in more detailed information about regular expressions and their variants (POSIX regex *etc.*) have a look at the following dedicated book on this topic:

Mastering Regular Expressions, 2nd Edition Jeffrey E.F. Friedl O'Reilly & Associates, Inc. 2002 ISBN 0-596-00289-0

Additionally in mod*rewrite the NOT character ('* `!` *') is a possible pattern prefix. This gives you the ability to negate a pattern; to say, for instance: "_if the current URL does* ***NOT*** *match this pattern*". This can be used for exceptional cases, where it is easier to match the negative pattern, or as a last default rule.

Notice

When using the NOT character to negate a pattern you cannot have grouped wildcard parts in the pattern. This is impossible because when the pattern does NOT match, there are no contents for the groups. In consequence, if negated patterns are used, you cannot use `$N` in the substitution string!

Substitution of a rewriting rule is the string which is substituted for (or replaces) the original URL for which *Pattern* matched. Beside plain text you can use

1. back-references `$N` to the RewriteRule pattern
2. back-references `%N` to the last matched RewriteCond pattern
3. server-variables as in rule condition test-strings (`%{VARNAME}`)
4. mapping-function calls (`${mapname:key|default}`)

Back-references are `$` **N** (**N**=0..9) identifiers which will be replaced by the contents of the **N**th group of the matched *Pattern*. The server-variables are the same as for the *TestString* of a `RewriteCond` directive. The mapping-functions come from the `RewriteMap` directive and are explained there. These three types of variables are expanded in the order of the above list.

As already mentioned above, all the rewriting rules are applied to the *Substitution* (in the order of definition in the config file). The URL is **completely replaced** by the *Substitution* and the rewriting process goes on until there are no more rules unless explicitly terminated by a `**L**` flag - see below.

There is a special substitution string named '`-`' which means: **NO substitution**! Sounds silly? No, it is useful to provide rewriting rules which **only** match some URLs but do no substitution, 例如， in conjunction with the **C** (chain) flag to be able to have more than one pattern to be applied before a substitution occurs.

Query String

Pattern will not match against the query string. Instead, you must use a `RewriteCond` with the `%{QUERY_STRING}` variable. You can, however, create URLs in the substitution string containing a query string part. Just use a question mark inside the substitution string to indicate that the following stuff should be re-injected into the query string. When you want to erase an existing query string, end the substitution string with just the question mark. To combine a new query string with an old one, use the `[QSA]` flag (see below).

Substitution of Absolute URLs

There is a special feature: When you prefix a substitution field with `http://` *thishost*[*:thisport*] then **mod_rewrite** automatically strips it out. This auto-reduction on implicit external redirect URLs is a useful and important feature when used in combination with a mapping-function which generates the hostname part. Have a look at the first example in the example section below to understand this.

Remember: An unconditional external redirect to your own server will not work with the prefix `http://thishost` because of this feature. To achieve such a self-redirect, you have to use the **R**-flag (see below).

Additionally you can set special flags for *Substitution* by appending

`[` ***flags*** `]`

as the third argument to the `RewriteRule` directive. *Flags* is a comma-separated list of the following flags:

- '`chain|C`' (**c**hained with next rule) This flag chains the current rule with the next rule (which itself can be chained with the following rule, *etc.*). This has the following effect: if a rule matches, then processing continues as usual, *i.e.*, the flag has no effect. If the rule does **not** match, then all following chained rules are skipped. For instance, use it to remove the "`.www`" part inside a per-directory rule set when you let an external redirect happen (where the "`.www`" part should not to occur!).
- '`cookie|CO=`*NAME*:*VAL*:*domain*[:*lifetime*[:*path*]]' (set **co**okie) This sets a cookie on the client's browser. The cookie's name is specified by *NAME* and the value is *VAL*. The *domain* field is the domain of the cookie, such as '.apache.org',the optional *lifetime* is the lifetime of the cookie in minutes, and the optional *path* is the path of the cookie
- '`env|E=`*VAR*:*VAL*' (set **e**nvironment variable) This forces an environment variable named *VAR* to be set to the value *VAL*, where *VAL* can contain regexp backreferences `$N` 和 `%N` which will be expanded. You can use this flag more than once to set more than one variable. The variables can be later dereferenced in many situations, but usually from within XSSI (via `<!--#echo var="VAR"-->`) or CGI (例如， `$ENV{'VAR'}`). Additionally you can dereference it in a following RewriteCond pattern via `%{ENV:VAR}`. Use this to strip but remember information from URLs.
- '`forbidden|F`' (force URL to be **f**orbidden) This forces the current URL to be forbidden, *i.e.*, it immediately sends back a HTTP response of 403 (FORBIDDEN). Use this flag in conjunction with appropriate RewriteConds to conditionally block some URLs.
- '`gone|G`' (force URL to be **g**one) This forces the current URL to be gone, *i.e.*, it immediately sends back a HTTP response of 410 (GONE). Use this flag to mark pages which no longer exist as gone.
- '`handler|H`=*Content-handler*' (force Content **h**andler) Force the Content-handler of the target file to be *Content-handler*. For instance, this can be used to simulate the `mod_alias` directive `ScriptAlias` which internally forces all files inside the mapped directory to have a handler of "`cgi-script`".
- '`last|L`' (**l**ast rule) Stop the rewriting process here and don't apply any more rewriting rules. This corresponds to the Perl `last` command or the `break` command from the C language. Use this flag to prevent the currently rewritten URL from being rewritten further by following rules. For example, use it to rewrite the root-path URL ('`/`') to a real one, 例如，'`/e/www/`'.
- '`next|N`' (**n**ext round) Re-run the rewriting process (starting again with the first rewriting rule). Here the URL to match is again not the original URL but the URL from the last rewriting rule. This corresponds to the Perl `next` command or the `continue` command from the C language. Use this flag to restart the rewriting process, *i.e.*, to immediately go to the top of the loop. **But be careful not to create an infinite loop!**
- '`nocase|NC`' (**n**o **c**ase) This makes the *Pattern* case-insensitive, *i.e.*, there is no difference between 'A-Z' and 'a-z' when *Pattern* is matched against the current URL.

- '**`noescape|NE`**' (**n**o URI **e**scaping of output) This flag keeps mod_rewrite from applying the usual URI escaping rules to the result of a rewrite. Ordinarily, special characters (such as '%', '$', ';', and so on) will be escaped into their hexcode equivalents ('%25', '%24', and '%3B', respectively); this flag prevents this from being done. This allows percent symbols to appear in the output, as in

  ```
  RewriteRule /foo/(.*) /bar?arg=P1\%3d$1 [R,NE]
  ```

  which would turn '`/foo/zed`' into a safe request for '`/bar?arg=P1=zed`'.

- '**`nosubreq|NS`**' (used only if **n**o internal **s**ub-request) This flag forces the rewriting engine to skip a rewriting rule if the current request is an internal sub-request. For instance, sub-requests occur internally in Apache when `mod_include` tries to find out information about possible directory default files (`index.xxx`). On sub-requests it is not always useful and even sometimes causes a failure to if the complete set of rules are applied. Use this flag to exclude some rules.

  Use the following rule for your decision: whenever you prefix some URLs with CGI-scripts to force them to be processed by the CGI-script, the chance is high that you will run into problems (or even overhead) on sub-requests. In these cases, use this flag.

- '**`proxy|P`**' (force **p**roxy) This flag forces the substitution part to be internally forced as a proxy request and immediately (*i.e.*, rewriting rule processing stops here) put through the proxy module. You have to make sure that the substitution string is a valid URI (例如，typically starting with `http://` *hostname*) which can be handled by the Apache proxy module. If not you get an error from the proxy module. Use this flag to achieve a more powerful implementation of the ProxyPass directive, to map some remote stuff into the namespace of the local server.

  注意： `mod_proxy` must be enabled in order to use this flag.

- '**`passthrough|PT`**' (**pass t**hrough to next handler) This flag forces the rewriting engine to set the `uri` field of the internal `request_rec` structure to the value of the `filename` field. This flag is just a hack to be able to post-process the output of `RewriteRule` directives by `Alias` , `ScriptAlias` , `Redirect` , *etc.* directives from other URI-to-filename translators. A trivial example to show the semantics: If you want to rewrite `/abc` to `/def` via the rewriting engine of `mod_rewrite` and then `/def` to `/ghi` with `mod_alias` :

  ```
  RewriteRule ^/abc(.*)  /def$1 [PT]
  Alias       /def       /ghi
  ```

If you omit the `PT` flag then `mod_rewrite` will do its job fine, *i.e.*, it rewrites `uri=/abc/...` to `filename=/def/...` as a full API-compliant URI-to-filename translator should do. Then `mod_alias` comes and tries to do a URI-to-filename transition which will not work.

Note: **You have to use this flag if you want to intermix directives of different modules which contain URL-to-filename translators**. The typical example is the use of `mod_alias` 和 `mod_rewrite` ..

- '**`qsappend|QSA`**' (**q**uery **s**tring **a**ppend) This flag forces the rewriting engine to append a query string part in the substitution string to the existing one instead of replacing it. Use this when you want to add more data to the query string via a rewrite rule.
- '**`redirect|R`** **[=*code*]**' (force **r**edirect) Prefix *Substitution* with `http://thishost[:thisport]/` (which makes the new URL a URI) to force a external redirection. If no *code* is given a HTTP response of 302 (MOVED TEMPORARILY) is used. If you want to use other response codes in the range 300-400 just specify them as a number or use one of the following symbolic names: `temp` (default), `permanent`, `seeother`. Use it for rules which should canonicalize the URL and give it back to the client, 例如, translate "`/~`" into "`/u/`" or always append a slash to `/u/` *user*, etc.

  Note: When you use this flag, make sure that the substitution field is a valid URL! If not, you are redirecting to an invalid location! And remember that this flag itself only prefixes the URL with `http://thishost[:thisport]/`, rewriting continues. Usually you also want to stop and do the redirection immediately. To stop the rewriting you also have to provide the 'L' flag.
- '**`skip|S`** =*num*' (**s**kip next rule(s)) This flag forces the rewriting engine to skip the next *num* rules in sequence when the current rule matches. Use this to make pseudo if-then-else constructs: The last rule of the then-clause becomes `skip=N` where N is the number of rules in the else-clause. (This is **not** the same as the 'chain|C' flag!)
- '**`type|T`** =*MIME-type*' (force MIME **t**ype) Force the MIME-type of the target file to be *MIME-type*. For instance, this can be used to setup the content-type based on some conditions. For example, the following snippet allows `.php` files to be *displayed* by `mod_php` if they are called with the `.phps` extension:

  ```
  RewriteRule ^(.+\.php)s$ $1 [T=application/x-httpd-php-source]
  ```

注意

Never forget that *Pattern* is applied to a complete URL in per-server configuration files. **But in per-directory configuration files, the per-directory prefix (which always is the same for a specific directory!) is automatically *removed* for the pattern matching and automatically *added* after the substitution has been done.** This feature is essential for many sorts of rewriting, because without this prefix stripping you have to match the parent directory which is not always possible.

There is one exception: If a substitution string starts with "`http://`" then the directory prefix will **not** be added and an external redirect or proxy throughput (if flag **P** is used!) is forced!

注意

To enable the rewriting engine for per-directory configuration files you need to set "`RewriteEngine On`" in these files 和 "`Options FollowSymLinks`" must be enabled. If your administrator has disabled override of `FollowSymLinks` for a user's directory, then you cannot use the rewriting engine. This restriction is needed for security reasons.

Here are all possible substitution combinations and their meanings:

Inside per-server configuration (`httpd.conf`) for request "`GET /somepath/pathinfo`":

```
**Given Rule**                                      **Resulting Substitution**
----------------------------------------------  ----------------------------------
^/somepath(.*) otherpath$1                      not supported, because invalid!

^/somepath(.*) otherpath$1  [R]                 not supported, because invalid!

^/somepath(.*) otherpath$1  [P]                 not supported, because invalid!
----------------------------------------------  ----------------------------------
^/somepath(.*) /otherpath$1                     /otherpath/pathinfo

^/somepath(.*) /otherpath$1 [R]                 http://thishost/otherpath/pathinfo
                                                via external redirection

^/somepath(.*) /otherpath$1 [P]                 not supported, because silly!
----------------------------------------------  ----------------------------------
^/somepath(.*) http://thishost/otherpath$1      /otherpath/pathinfo

^/somepath(.*) http://thishost/otherpath$1 [R]  http://thishost/otherpath/pathinfo
                                                via external redirection

^/somepath(.*) http://thishost/otherpath$1 [P]  not supported, because silly!
----------------------------------------------  ----------------------------------
^/somepath(.*) http://otherhost/otherpath$1     http://otherhost/otherpath/pathinfo
                                                via external redirection

^/somepath(.*) http://otherhost/otherpath$1 [R] http://otherhost/otherpath/pathinfo
                                                via external redirection
                                                (the [R] flag is redundant)

^/somepath(.*) http://otherhost/otherpath$1 [P] http://otherhost/otherpath/pathinfo
                                                via internal proxy
```

Inside per-directory configuration for `/somepath` (*i.e.*, file `.htaccess` in dir `/physical/path/to/somepath` containing `RewriteBase /somepath`) for request "`GET /somepath/localpath/pathinfo`":

```
**Given Rule**                                      **Resulting Substitution**
----------------------------------------------  ----------------------------------
^localpath(.*) otherpath$1                      /somepath/otherpath/pathinfo

^localpath(.*) otherpath$1  [R]                 http://thishost/somepath/otherpath/pathin
                                                via external redirection

^localpath(.*) otherpath$1  [P]                 not supported, because silly!
----------------------------------------------  ----------------------------------
^localpath(.*) /otherpath$1                     /otherpath/pathinfo

^localpath(.*) /otherpath$1 [R]                 http://thishost/otherpath/pathinfo
                                                via external redirection

^localpath(.*) /otherpath$1 [P]                 not supported, because silly!
----------------------------------------------  ----------------------------------
^localpath(.*) http://thishost/otherpath$1      /otherpath/pathinfo

^localpath(.*) http://thishost/otherpath$1 [R]  http://thishost/otherpath/pathinfo
                                                via external redirection

^localpath(.*) http://thishost/otherpath$1 [P]  not supported, because silly!
----------------------------------------------  ----------------------------------
^localpath(.*) http://otherhost/otherpath$1     http://otherhost/otherpath/pathinfo
                                                via external redirection

^localpath(.*) http://otherhost/otherpath$1 [R] http://otherhost/otherpath/pathinfo
                                                via external redirection
                                                (the [R] flag is redundant)

^localpath(.*) http://otherhost/otherpath$1 [P] http://otherhost/otherpath/pathinfo
                                                via internal proxy
```

Example:

We want to rewrite URLs of the form

`/` *Language* `/~` *Realname* `/.../` *File*

into

`/u/` *Username* `/.../` *File* `.` *Language*

We take the rewrite mapfile from above and save it under `/path/to/file/map.txt`. Then we only have to add the following lines to the Apache server configuration file:

```
RewriteLog   /path/to/file/rewrite.log
RewriteMap   real-to-user               txt:/path/to/file/map.txt
RewriteRule  ^/([^/]+)/~([^/]+)/(.*)$   /u/${real-to-user:$2|nobody}/$3.$1
```

Apache模块 mod_setenvif

说明	根据客户端请求头字段设置环境变量
状态	基本(B)
模块名	setenvif_module
源文件	mod_setenvif.c

概述

`mod_setenvif` 模块允许根据请求的不同方面匹配指定的正则表达式来设置环境变量。这些环境变量可由服务器的其他部分使用。

指令按照他们在配置文件中出现的顺序生效。所以可以使用更多的复合序列，正如下例所示，如果浏览器是mozilla而非MSIE则会设置 `netscape` 。

```
BrowserMatch ^Mozilla netscape
BrowserMatch MSIE !netscape
```

BrowserMatch 指令

说明	基于**User-Agent**头有条件地设置环境变量
语法	`BrowserMatch _regex [!]env-variable_[=_value_] [[!]_env-variable_[=_value_]] ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_setenvif

`BrowserMatch` 只是 `SetEnvIf` 的一种特殊情况，基于 `User-Agent` 头有条件地设置环境变量。下面的两行具有相同的效果：

```
BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot
```

更多的例子：

```
BrowserMatch ^Mozilla forms jpeg=yes browser=netscape
BrowserMatch "^Mozilla/[2-3]" tables agif frames javascript
BrowserMatch MSIE !javascript
```

BrowserMatchNoCase 指令

说明	基于不区分大小写的**User-Agent**头有条件地设置环境变量
语法	`BrowserMatchNoCase _regex [!]env-variable_[=_value_] [[!]_env-variable_[=_value_]]`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_setenvif

`BrowserMatchNoCase` 等同于 `BrowserMatch` ，仅仅是进行不区分大小写的匹配。例如：

```
BrowserMatchNoCase mac platform=macintosh
BrowserMatchNoCase win platform=windows
```

`BrowserMatch` 和 `BrowserMatchNoCase` 只是 `SetEnvIf` 和 `SetEnvIfNoCase` 的一种特殊情况。下面的两行具有相同的效果：

```
BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot
```

SetEnvIf 指令

说明	根据客户端请求属性设置环境变量
语法	`SetEnvIf _attribute regex [!]env-variable_[=_value_] [[!]_env-variable_[=_value_]]`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_setenvif

`SetEnvIf` 指令根据客户端的请求属性设置环境变量。第一个参数*attribute*必须是下列三种类别之一：

1. 一个HTTP请求头域(参见RFC2616)；例如：`Host` , `User-Agent` , `Referer` , `Accept-Language` 。可以用一个正则表达式来进行匹配。
2. 下列请求属性之一：
   - `Remote_Host` 远程主机名(若可用)
   - `Remote_Addr` 远程主机IP地址
   - `Server_Addr` 接收到请求的服务器IP地址(2.0.43及以后版本)
   - `Request_Method` 所用的请求方法(`GET` , `POST` 等等)
   - `Request_Protocol` 请求所使用的协议及其版本("HTTP/0.9", "HTTP/1.0", "HTTP/1.1"等)
   - `Request_URI` 在HTTP请求行中请求的资源(通常是URL中去除协议、主机以及查询字符串后剩余的部分)。
3. 列出的与该请求关联的环境变量名字中的一个。这将允许 `SetEnvIf` 指令基于预先匹配的结果进行测试。只有那些由较早的 `SetEnvIf[NoCase]` 指令定义的环境变量才可以按照这种方式测试。"较早的"意思是它们在更上层的作用域(比如全局范围)中被定义或者在同一作用域中较早出现。只有在请求的属性未能得到匹配并且*attribute*没有使用正则表达式的时候，环境变量才会被考虑。

第二个参数*regex*是一个Perl兼容的正则表达式。如果*regex*是根据*attribute*进行匹配的，那么剩余的参数将被评估。剩余的参数给出了需要设置的变量名及其可选的值。格式如下：

1. `_varname_`
2. `!_varname_`
3. `_varname_=_value_`

第一个格式，环境变量 `_varname_` 的值将设为"1"。第二个格式将删除给定的变量 `_varname_` (若存在)。第三个格式将为环境变量 `_varname_` 设置 `_value_` 的字面值。从2.0.51版开始，Apache能够识别value中出现的 `$1` .. `$9` ，并将其替换为regex中对应的使用括号括起来的子模式。

示例：

```
SetEnvIf Request_URI "\.gif$" object_is_image=gif
SetEnvIf Request_URI "\.jpg$" object_is_image=jpg
SetEnvIf Request_URI "\.xbm$" object_is_image=xbm
     :
SetEnvIf Referer www\.mydomain\.com intra_site_referral
     :
SetEnvIf object_is_image xbm XBIT_PROCESSING=1
     :
SetEnvIf ^TS*  ^[a-z].*  HAVE_TS
```

前面的三个将会设置 `object_is_image` 环境变量(如果请求的是图片)。第四个将会设置 `intra_site_referral` 环境变量(如果Referer头表明来自于 `www.mydomain.com`)。

最后一个将会设置环境变量 `HAVE_TS` (如果包含任何以"TS"开始的请求头，并且该请求头的值是以小写字母[a-z]开头的)。

参见

- Apache环境变量，以获得更多例子。

SetEnvIfNoCase 指令

说明	根据大小写无关的客户端请求属性设置环境变量
语法	`SetEnvIfNoCase _attribute regex [!]env-variable_[=_value_] [[!]_env-variable_[=_va`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	基本(B)
模块	mod_setenvif

`SetEnvIfNoCase` 等同于 `SetEnvIf` ，仅仅是进行不区分大小写的匹配。例如：

```
SetEnvIfNoCase Host Apache\.Org site=apache
```

这会将环境变量 `site` 设置为" `apache` "(如果" `Host:` "请求头包含" `Apache.Org` "、" `apache.org` "等大小写变体)。

Apache模块 mod_so

说明	允许运行时加载DSO模块
状态	扩展(E)
模块名	so_module
源文件	mod_so.c
兼容性	在Windows上这是一个基本模块(始终包含)

概述

该模块在支持动态链接机制的操作系统上可以用来在Apache启动和重启时加载DSO模块，而不用重新编译。

在Unix上，被加载的可执行代码通常来自于共享对象文件(一般以 `.so` 为扩展名)，在Windows上则为 `.so` 或 `.dll` 扩展名。

警告

用于Apache1.3的模块不能直接用于Apache2.0 。

为Windows创建可加载模块

注意

从Apache1.3.15开始，Windows上的模块名规则发生了变化，现在使用mod_foo.so格式的名字。

Apache模块的API对于Unix和Windows是一样的。许多模块在这两个平台间移植几乎不需要什么修改，除非那些依赖于Unix特定属性而Windows没有提供的模块。

如果一个模块可用，那么有两种方法使用它。在Unix上，可以被静态编译进服务器。因为用于Windows的Apache并没有相应于Unix下的 `Configure` 编译配置程序，模块的源文件必须被加进ApacheCore项目文件，并且它的符号(symbols)必须被添加到 `os\win32\modules.c` 文件。

第二种方法是编译为一个动态链接库(DLL)，以便在运行期间使用 `LoadModule` 指令加载。这些模块DLL在Apache安装期间就已经安装好了，不需要你自己去编译。

为了将模块编译为DLL，需要对模块的源文件做一个小小的修改：模块记录(module record)必须从DLL导出(稍后将会创建，见下)。为了达到这个目的，请将 `AP_MODULE_DECLARE_DATA` (在Apache头文件中定义的)添加到你的模块记录(module record)定义中。比如，如果你的模块有：

```
    module foo_module;
```

将上述内容替换为：

```
    module AP_MODULE_DECLARE_DATA foo_module;
```

注意，这仅在Windows上有效，因此该模块可以不加修改的直接在Unix上使用。另外，如果你对 `.DEF` 文件很熟悉，你也可以使用它代替前面的方法导出该模块。

要创建一个包含该模块的DLL文件，你还必须将它连接到在编译libhttpd.dll共享库时创建的libhttpd.lib导出库。你还可能需要修改编译器设置以确保Apache头文件被正确的加载了。这些库位于服务器根目录下的"modules"目录中。最好是从中抓出一个已经存在的模块.dsp文件来看看以确保编译环境配置无误，或者按照.dsp文检查编译器和连接器的选项也可。

这样将会为你的模块创建一个DLL版本。只要将它放置到 `modules` 目录下，并使用 `LoadModule` 指令加载即可。

LoadFile 指令

说明	加载已命名的目标文件或库
语法	`LoadFile _filename_ [_filename_] ...`
作用域	server config
状态	扩展(E)
模块	mod_so

该指令用于在服务器启动或者重启时加载已命名目标文件或库，以用于加载需要被某些模块使用的额外代码。*Filename*可以是一个绝对路径或者相对于 `ServerRoot` 的相对路径。

例如：

```
LoadFile libexec/libxmlparse.so
```

LoadModule 指令

说明	加载目标文件或库，并将其添加到活动模块列表
语法	`LoadModule _module filename_`
作用域	server config
状态	扩展(E)
模块	mod_so

该指令加载目标文件或库*filename*并将模块结构名*module*添加到活动模块列表。*module*就是源代码文件中用于拼写 `module` 的外部变量名，并作为模块标识符(Module Identifier)列在模块文档中。例如：

```
LoadModule status_module modules/mod_status.so
```

加载了位于 `ServerRoot` 下模块目录中指定的模块。

Apache模块 mod_speling

说明	自动纠正URL中的拼写错误
状态	扩展(E)
模块名	speling_module
源文件	mod_speling.c

概述

Requests to documents sometimes cannot be served by the core apache server because the request was misspelled or miscapitalized. This module addresses this problem by trying to find a matching document, even after all other modules gave up. It does its work by comparing each document name in the requested directory against the requested document name **without regard to case**, and allowing **up to one misspelling** (character insertion / omission / transposition or wrong character). A list is built with all document names which were matched using this strategy.

If, after scanning the directory,

- no matching document was found, Apache will proceed as usual and return a "document not found" error.
- only one document is found that "almost" matches the request, then it is returned in the form of a redirection response.
- more than one document with a close match was found, then the list of the matches is returned to the client, and the client can select the correct candidate.

CheckSpelling 指令

说明	**Enables the spelling module**
语法	`CheckSpelling on|off`
默认值	`CheckSpelling Off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Options
状态	扩展(E)
模块	mod_speling
兼容性	CheckSpelling was available as a separately available module for Apache 1.1, but was limited to miscapitalizations. As of Apache 1.3, it is part of the Apache distribution. Prior to Apache 1.3.2, the `CheckSpelling` directive was only available in the "server" and "virtual host" contexts.

This directive enables or disables the spelling module. When enabled, keep in mind that

- the directory scan which is necessary for the spelling correction will have an impact on the server's performance when many spelling corrections have to be performed at the same time.
- the document trees should not contain sensitive files which could be matched inadvertently by a spelling "correction".
- the module is unable to correct misspelled user names (as in `http://my.host/~apahce/`), just file names or directory names.
- spelling corrections apply strictly to existing files, so a request for the `<Location /status>` may get incorrectly treated as the negotiated file " `/stats.html` ".

mod_speling should not be enabled in DAV enabled directories, because it will try to "spell fix" newly created resource names against existing filenames, e.g., when trying to upload a new document `doc43.html` it might redirect to an existing document `doc34.html` , which is not what was intended.

Apache模块 mod_ssl

说明	使用安全套接字层(SSL)和传输层安全(TLS)协议实现高强度加密传输
状态	扩展(E)
模块名	ssl_module
源文件	mod_ssl.c

概述

This module provides SSL v2/v3 and TLS v1 support for the Apache HTTP Server. It was contributed by Ralf S. Engeschall based on his mod_ssl project and originally derived from work by Ben Laurie.

This module relies on OpenSSL to provide the cryptography engine.

Further details, discussion, and examples are provided in the SSL documentation.

Environment Variables

This module provides a lot of SSL information as additional environment variables to the SSI and CGI namespace. The generated variables are listed in the table below. For backward compatibility the information can be made available under different names, too. Look in the Compatibility chapter for details on the compatibility variables.

Variable Name:	Value Type:	Description:
`HTTPS`	flag	HTTPS is being used.
`SSL_PROTOCOL`	string	The SSL protocol version (SSLv2, SSLv3, TLSv1)
`SSL_SESSION_ID`	string	The hex-encoded SSL session id
`SSL_CIPHER`	string	The cipher specification name
`SSL_CIPHER_EXPORT`	string	`true` if cipher is an export cipher
`SSL_CIPHER_USEKEYSIZE`	number	Number of cipher bits (actually used)
`SSL_CIPHER_ALGKEYSIZE`	number	Number of cipher bits (possible)

`SSL_COMPRESS_METHOD`	string	SSL compression method negotiated
`SSL_VERSION_INTERFACE`	string	The mod_ssl program version
`SSL_VERSION_LIBRARY`	string	The OpenSSL program version
`SSL_CLIENT_M_VERSION`	string	The version of the client certificate
`SSL_CLIENT_M_SERIAL`	string	The serial of the client certificate
`SSL_CLIENT_S_DN`	string	Subject DN in client's certificate
`SSL_CLIENT_S_DN_` *x509*	string	Component of client's Subject DN
`SSL_CLIENT_I_DN`	string	Issuer DN of client's certificate
`SSL_CLIENT_I_DN_` *x509*	string	Component of client's Issuer DN
`SSL_CLIENT_V_START`	string	Validity of client's certificate (start time)
`SSL_CLIENT_V_END`	string	Validity of client's certificate (end time)
`SSL_CLIENT_V_REMAIN`	string	Number of days until client's certificate expires
`SSL_CLIENT_A_SIG`	string	Algorithm used for the signature of client's certificate
`SSL_CLIENT_A_KEY`	string	Algorithm used for the public key of client's certificate
`SSL_CLIENT_CERT`	string	PEM-encoded client certificate
`SSL_CLIENT_CERT_CHAIN_` *n*	string	PEM-encoded certificates in client certificate chain
`SSL_CLIENT_VERIFY`	string	`NONE` , `SUCCESS` , `GENEROUS` 或 `FAILED:` *reason*
`SSL_SERVER_M_VERSION`	string	The version of the server certificate
`SSL_SERVER_M_SERIAL`	string	The serial of the server certificate
`SSL_SERVER_S_DN`	string	Subject DN in server's certificate
`SSL_SERVER_S_DN_` *x509*	string	Component of server's Subject DN

SSL_SERVER_I_DN	string	Issuer DN of server's certificate
SSL_SERVER_I_DN_ *x509*	string	Component of server's Issuer DN
SSL_SERVER_V_START	string	Validity of server's certificate (start time)
SSL_SERVER_V_END	string	Validity of server's certificate (end time)
SSL_SERVER_A_SIG	string	Algorithm used for the signature of server's certificate
SSL_SERVER_A_KEY	string	Algorithm used for the public key of server's certificate
SSL_SERVER_CERT	string	PEM-encoded server certificate

x509 specifies a component of an X.509 DN; one of `C,ST,L,O,OU,CN,T,I,G,S,D,UID,Email` . In Apache 2.1 and later, *x509* may also include a numeric `_n` suffix. If the DN in question contains multiple attributes of the same name, this suffix is used as an index to select a particular attribute. For example, where the server certificate subject DN included two OU fields, `SSL_SERVER_S_DN_OU_0` 和 `SSL_SERVER_S_DN_OU_1` could be used to reference each.

`SSL_CLIENT_V_REMAIN` is only available in version 2.1 and later.

Custom Log Formats

When `mod_ssl` is built into Apache or at least loaded (under DSO situation) additional functions exist for the Custom Log Format of `mod_log_config` . First there is an additional " `%{` *varname* `}x` " eXtension format function which can be used to expand any variables provided by any module, especially those provided by mod_ssl which can you find in the above table.

For backward compatibility there is additionally a special " `%{` *name* `}c` " cryptography format function provided. Information about this function is provided in the Compatibility chapter.

示例

```
CustomLog logs/ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
```

SSLCACertificateFile 指令

说明	File of concatenated PEM-encoded CA Certificates for Client Auth
语法	`SSLCACertificateFile _file-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificates of Certification Authorities (CA) whose *clients* you deal with. These are used for Client Authentication. Such a file is simply the concatenation of the various PEM-encoded Certificate files, in order of preference. This can be used alternatively and/or additionally to `SSLCACertificatePath` .

示例

```
SSLCACertificateFile /usr/local/apache2/conf/ssl.crt/ca-bundle-client.crt
```

SSLCACertificatePath 指令

说明	Directory of PEM-encoded CA Certificates for Client Auth
语法	`SSLCACertificatePath _directory-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive sets the directory where you keep the Certificates of Certification Authorities (CAs) whose clients you deal with. These are used to verify the client certificate on Client Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value* `.N` . And you should always make sure this directory contains the appropriate symbolic links. Use the `Makefile` which comes with mod_ssl to accomplish this task.

示例

```
SSLCACertificatePath /usr/local/apache2/conf/ssl.crt/
```

SSLCADNRequestFile 指令

说明	File of concatenated PEM-encoded CA Certificates for defining acceptable CA names
语法	`SSLCADNRequestFile _file-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

When a client certificate is requested by mod*ssl, a list of _acceptable Certificate Authority names* is sent to the client in the SSL handshake. These CA names can be used by the client to select an appropriate client certificate out of those it has available.

If neither of the directives `SSLCADNRequestPath` 或 `SSLCADNRequestFile` are given, then the set of acceptable CA names sent to the client is the names of all the CA certificates given by the `SSLCACertificateFile` 和 `SSLCACertificatePath` directives; in other words, the names of the CAs which will actually be used to verify the client certificate.

In some circumstances, it is useful to be able to send a set of acceptable CA names which differs from the actual CAs used to verify the client certificate - for example, if the client certificates are signed by intermediate CAs. In such cases, `SSLCADNRequestPath` and/or `SSLCADNRequestFile` can be used; the acceptable CA names are then taken from the complete set of certificates in the directory and/or file specified by this pair of directives.

`SSLCADNRequestFile` must specify an *all-in-one* file containing a concatenation of PEM-encoded CA certificates.

示例

```
SSLCADNRequestFile /usr/local/apache2/conf/ca-names.crt
```

SSLCADNRequestPath 指令

说明	Directory of PEM-encoded CA Certificates for defining acceptable CA names
语法	`SSLCADNRequestPath _directory-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This optional directive can be used to specify the set of *acceptable CA names* which will be sent to the client when a client certificate is requested. See the `SSLCADNRequestFile` directive for more details.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value* `.N`. And you should always make sure this directory contains the appropriate symbolic links. Use the `Makefile` which comes with mod_ssl to accomplish this task.

示例

```
SSLCADNRequestPath /usr/local/apache2/conf/ca-names.crt/
```

SSLCARevocationFile 指令

说明	File of concatenated PEM-encoded CA CRLs for Client Auth
语法	`SSLCARevocationFile _file-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose *clients* you deal with. These are used for Client Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to `SSLCARevocationPath`.

示例

```
SSLCARevocationFile /usr/local/apache2/conf/ssl.crl/ca-bundle-client.crl
```

SSLCARevocationPath 指令

说明	Directory of PEM-encoded CA CRLs for Client Auth
语法	`SSLCARevocationPath _directory-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive sets the directory where you keep the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) whose clients you deal with. These are used to revoke the client certificate on Client Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you have not only to place the CRL files there. Additionally you have to create symbolic links named *hash-value* `.rN` . And you should always make sure this directory contains the appropriate symbolic links. Use the `Makefile` which comes with `mod_ssl` to accomplish this task.

示例

```
SSLCARevocationPath /usr/local/apache2/conf/ssl.crl/
```

SSLCertificateChainFile 指令

说明	File of PEM-encoded Server CA Certificates
语法	`SSLCertificateChainFile _file-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive sets the optional *all-in-one* file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate. This starts with the issuing CA certificate of of the server certificate and can range up to the root CA certificate. Such a file is simply the concatenation of the various PEM-encoded CA Certificate files, usually in certificate chain order.

This should be used alternatively and/or additionally to `SSLCACertificatePath` for explicitly constructing the server certificate chain which is sent to the browser in addition to the server certificate. It is especially useful to avoid conflicts with CA certificates when using client authentication. Because although placing a CA certificate of the server certificate chain into `SSLCACertificatePath` has the same effect for the certificate chain construction, it has the side-effect that client certificates issued by this same CA certificate are also accepted on client authentication. That's usually not one expect.

But be careful: Providing the certificate chain works only if you are using a *single* (either RSA 或 DSA) based server certificate. If you are using a coupled RSA+DSA certificate pair, this will work only if actually both certificates use the *same* certificate chain. Else the browsers will be confused in this situation.

示例

```
SSLCertificateChainFile /usr/local/apache2/conf/ssl.crt/ca.crt
```

SSLCertificateFile 指令

说明	Server PEM-encoded X.509 Certificate file
语法	`SSLCertificateFile _file-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive points to the PEM-encoded Certificate file for the server and optionally also to the corresponding RSA or DSA Private Key file for it (contained in the same file). If the contained Private Key is encrypted the Pass Phrase dialog is forced at startup time. This directive can be used up to two times (referencing different filenames) when both a RSA and a DSA based server certificate is used in parallel.

示例

```
SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
```

SSLCertificateKeyFile 指令

说明	Server PEM-encoded Private Key file
语法	`SSLCertificateKeyFile _file-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive points to the PEM-encoded Private Key file for the server. If the Private Key is not combined with the Certificate in the `SSLCertificateFile`, use this additional directive to point to the file with the stand-alone Private Key. When `SSLCertificateFile` is used and the file contains both the Certificate and the Private Key this directive need not be used. But we strongly discourage this practice. Instead we recommend you to separate the Certificate and the Private Key. If the contained Private Key is encrypted, the Pass Phrase dialog is forced at startup time. This directive can be used up to two times (referencing different filenames) when both a RSA and a DSA based private key is used in parallel.

示例

```
SSLCertificateKeyFile /usr/local/apache2/conf/ssl.key/server.key
```

SSLCipherSuite 指令

说明	Cipher Suite available for negotiation in SSL handshake
语法	`SSLCipherSuite _cipher-spec_`
默认值	`SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP`
作用域	server config, virtual host, directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_ssl

This complex directive uses a colon-separated *cipher-spec* string consisting of OpenSSL cipher specifications to configure the Cipher Suite the client is permitted to negotiate in the SSL handshake phase. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotation with the reconfigured Cipher Suite after the HTTP request was read but before the HTTP response is sent.

An SSL cipher specification in *cipher-spec* is composed of 4 major attributes plus a few extra minor ones:

- *Key Exchange Algorithm*: RSA or Diffie-Hellman variants.
- *Authentication Algorithm*: RSA, Diffie-Hellman, DSS or none.
- *Cipher/Encryption Algorithm*: DES, Triple-DES, RC4, RC2, IDEA or none.
- *MAC Digest Algorithm*: MD5, SHA or SHA1.

An SSL cipher can also be an export cipher and is either a SSLv2 or SSLv3/TLSv1 cipher (here TLSv1 is equivalent to SSLv3). To specify which ciphers to use, one can either specify all the Ciphers, one at a time, or use aliases to specify the preference and order for the ciphers (see Table 1).

Tag	**Description**
Key Exchange Algorithm:	
`kRSA`	RSA key exchange
`kDHr`	Diffie-Hellman key exchange with RSA key
`kDHd`	Diffie-Hellman key exchange with DSA key
`kEDH`	Ephemeral (temp.key) Diffie-Hellman key exchange (no cert)
Authentication Algorithm:	
`aNULL`	No authentication
`aRSA`	RSA authentication
`aDSS`	DSS authentication
`aDH`	Diffie-Hellman authentication
Cipher Encoding Algorithm:	
`eNULL`	No encoding
`DES`	DES encoding
`3DES`	Triple-DES encoding
`RC4`	RC4 encoding
`RC2`	RC2 encoding
`IDEA`	IDEA encoding
MAC Digest Algorithm:	
`MD5`	MD5 hash function
`SHA1`	SHA1 hash function

`SHA`	SHA hash function
Aliases:	
`SSLv2`	all SSL version 2.0 ciphers
`SSLv3`	all SSL version 3.0 ciphers
`TLSv1`	all TLS version 1.0 ciphers
`EXP`	all export ciphers
`EXPORT40`	all 40-bit export ciphers only
`EXPORT56`	all 56-bit export ciphers only
`LOW`	all low strength ciphers (no export, single DES)
`MEDIUM`	all ciphers with 128 bit encryption
`HIGH`	all ciphers using Triple-DES
`RSA`	all ciphers using RSA key exchange
`DH`	all ciphers using Diffie-Hellman key exchange
`EDH`	all ciphers using Ephemeral Diffie-Hellman key exchange
`ADH`	all ciphers using Anonymous Diffie-Hellman key exchange
`DSS`	all ciphers using DSS authentication
`NULL`	all ciphers using no encryption

Now where this becomes interesting is that these can be put together to specify the order and ciphers you wish to use. To speed this up there are also aliases (`SSLv2, SSLv3, TLSv1, EXP, LOW, MEDIUM, HIGH`) for certain groups of ciphers. These tags can be joined together with prefixes to form the *cipher-spec*. Available prefixes are:

- none: add cipher to list
- `+` : add ciphers to list and pull them to current location in list
- `-` : remove cipher from list (can be added later again)
- `!` : kill cipher from list completely (can **not** be added later again)

A simpler way to look at all of this is to use the " `openssl ciphers -v` " command which provides a nice way to successively create the correct *cipher-spec* string. The default *cipher-spec* string is " `ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP` " which means the following: first, remove from consideration any ciphers that do not authenticate, i.e. for SSL

only the Anonymous Diffie-Hellman ciphers. Next, use ciphers using RC4 and RSA. Next include the high, medium and then the low security ciphers. Finally *pull* all SSLv2 and export ciphers to the end of the list.

```
$ openssl ciphers -v 'ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP'
NULL-SHA                SSLv3 Kx=RSA      Au=RSA  Enc=None      Mac=SHA1
NULL-MD5                SSLv3 Kx=RSA      Au=RSA  Enc=None      Mac=MD5
EDH-RSA-DES-CBC3-SHA    SSLv3 Kx=DH       Au=RSA  Enc=3DES(168) Mac=SHA1
...                     ...               ...     ...           ...
EXP-RC4-MD5             SSLv3 Kx=RSA(512) Au=RSA  Enc=RC4(40)   Mac=MD5  export
EXP-RC2-CBC-MD5         SSLv2 Kx=RSA(512) Au=RSA  Enc=RC2(40)   Mac=MD5  export
EXP-RC4-MD5             SSLv2 Kx=RSA(512) Au=RSA  Enc=RC4(40)   Mac=MD5  export
```

The complete list of particular RSA & DH ciphers for SSL is given in Table 2.

示例

```
SSLCipherSuite RSA:!EXP:!NULL:+HIGH:+MEDIUM:-LOW
```

Cipher-Tag	Protocol	Key Ex.	Auth.	Enc.	MAC
RSA Ciphers:					
`DES-CBC3-SHA`	SSLv3	RSA	RSA	3DES(168)	SHA1
`DES-CBC3-MD5`	SSLv2	RSA	RSA	3DES(168)	MD5
`IDEA-CBC-SHA`	SSLv3	RSA	RSA	IDEA(128)	SHA1
`RC4-SHA`	SSLv3	RSA	RSA	RC4(128)	SHA1
`RC4-MD5`	SSLv3	RSA	RSA	RC4(128)	MD5
`IDEA-CBC-MD5`	SSLv2	RSA	RSA	IDEA(128)	MD5
`RC2-CBC-MD5`	SSLv2	RSA	RSA	RC2(128)	MD5
`RC4-MD5`	SSLv2	RSA	RSA	RC4(128)	MD5
`DES-CBC-SHA`	SSLv3	RSA	RSA	DES(56)	SHA1
`RC4-64-MD5`	SSLv2	RSA	RSA	RC4(64)	MD5
`DES-CBC-MD5`	SSLv2	RSA	RSA	DES(56)	MD5
`EXP-DES-CBC-SHA`	SSLv3	RSA(512)	RSA	DES(40)	SHA1
`EXP-RC2-CBC-MD5`	SSLv3	RSA(512)	RSA	RC2(40)	MD5
`EXP-RC4-MD5`	SSLv3	RSA(512)	RSA	RC4(40)	MD5
`EXP-RC2-CBC-MD5`	SSLv2	RSA(512)	RSA	RC2(40)	MD5

`EXP-RC4-MD5`	SSLv2	RSA(512)	RSA	RC4(40)	MD5
`NULL-SHA`	SSLv3	RSA	RSA	None	SHA1
`NULL-MD5`	SSLv3	RSA	RSA	None	MD5
Diffie-Hellman Ciphers:					
`ADH-DES-CBC3-SHA`	SSLv3	DH	None	3DES(168)	SHA1
`ADH-DES-CBC-SHA`	SSLv3	DH	None	DES(56)	SHA1
`ADH-RC4-MD5`	SSLv3	DH	None	RC4(128)	MD5
`EDH-RSA-DES-CBC3-SHA`	SSLv3	DH	RSA	3DES(168)	SHA1
`EDH-DSS-DES-CBC3-SHA`	SSLv3	DH	DSS	3DES(168)	SHA1
`EDH-RSA-DES-CBC-SHA`	SSLv3	DH	RSA	DES(56)	SHA1
`EDH-DSS-DES-CBC-SHA`	SSLv3	DH	DSS	DES(56)	SHA1
`EXP-EDH-RSA-DES-CBC-SHA`	SSLv3	DH(512)	RSA	DES(40)	SHA1
`EXP-EDH-DSS-DES-CBC-SHA`	SSLv3	DH(512)	DSS	DES(40)	SHA1
`EXP-ADH-DES-CBC-SHA`	SSLv3	DH(512)	None	DES(40)	SHA1
`EXP-ADH-RC4-MD5`	SSLv3	DH(512)	None	RC4(40)	MD5

SSLCryptoDevice 指令

说明	Enable use of a cryptographic hardware accelerator
语法	`SSLCryptoDevice _engine_`
默认值	`SSLCryptoDevice builtin`
作用域	server config
状态	扩展(E)
模块	mod_ssl
兼容性	Available if mod_ssl is built using `-DSSL_ENGINE_EXPERIMENTAL`

This directive enables use of a cryptographic hardware accelerator board to offload some of the SSL processing overhead. This directive can only be used if the SSL toolkit is built with "engine" support; OpenSSL 0.9.7 and later releases have "engine" support by default, the separate "-engine" releases of OpenSSL 0.9.6 must be used.

To discover which engine names are supported, run the command "`openssl engine`".

示例

```
# For a Broadcom accelerator:

SSLCryptoDevice ubsec
```

SSLEngine 指令

说明	SSL Engine Operation Switch
语法	`SSLEngine on|off|optional`
默认值	`SSLEngine off`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive toggles the usage of the SSL/TLS Protocol Engine. This is usually used inside a `<VirtualHost>` section to enable SSL/TLS for a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for both the main server and all configured virtual hosts.

示例

```
<VirtualHost _default_:443>

SSLEngine on

...

</VirtualHost>
```

In Apache 2.1 and later, `SSLEngine` can be set to `optional`. This enables support for RFC 2817, Upgrading to TLS Within HTTP/1.1. At this time no web browsers support RFC 2817.

SSLHonorCipherOrder 指令

说明	Option to prefer the server's cipher preference order
语法	`SSLHonorCiperOrder _flag_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl
兼容性	仅在 Apache 2.1 and later, if using OpenSSL 0.9.7 or later

When choosing a cipher during an SSLv3 or TLSv1 handshake, normally the client's preference is used. If this directive is enabled, the server's preference will be used instead.

示例

```
SSLHonorCipherOrder on
```

SSLMutex 指令

说明	Semaphore for internal mutual exclusion of operations
语法	`SSLMutex _type_`
默认值	`SSLMutex none`
作用域	server config
状态	扩展(E)
模块	mod_ssl

This configures the SSL engine's semaphore (aka. lock) which is used for mutual exclusion of operations which have to be done in a synchronized way between the pre-forked Apache server processes. This directive can only be used in the global server context because it's only useful to have one global mutex. This directive is designed to closely match the `AcceptMutex` directive.

The following Mutex *types* are available:

- `none | no`

  This is the default where no Mutex is used at all. Use it at your own risk. But because currently the Mutex is mainly used for synchronizing write access to the SSL Session Cache you can live without it as long as you accept a sometimes garbled Session Cache. So it's not recommended to leave this the default. Instead configure a real Mutex.

- `posixsem`

  This is an elegant Mutex variant where a Posix Semaphore is used when possible. It is only available when the underlying platform and APR supports it.

- `sysvsem`

This is a somewhat elegant Mutex variant where a SystemV IPC Semaphore is used when possible. It is possible to "leak" SysV semaphores if processes crash before the semaphore is removed. It is only available when the underlying platform and APR supports it.

- `sem`

  This directive tells the SSL Module to pick the "best" semaphore implementation available to it, choosing between Posix and SystemV IPC, in that order. It is only available when the underlying platform and APR supports at least one of the 2.

- `pthread`

  This directive tells the SSL Module to use Posix thread mutexes. It is only available if the underlying platform and APR supports it.

- `fcntl:/path/to/mutex`

  This is a portable Mutex variant where a physical (lock-)file and the `fcntl()` fucntion are used as the Mutex. Always use a local disk filesystem for `/path/to/mutex` and never a file residing on a NFS- or AFS-filesystem. It is only available when the underlying platform and APR supports it. Note: Internally, the Process ID (PID) of the Apache parent process is automatically appended to `/path/to/mutex` to make it unique, so you don't have to worry about conflicts yourself. Notice that this type of mutex is not available under the Win32 environment. There you *have* to use the semaphore mutex.

- `flock:/path/to/mutex`

  This is similar to the `fcntl:/path/to/mutex` method with the exception that the `flock()` function is used to provide file locking. It is only available when the underlying platform and APR supports it.

- `file:/path/to/mutex`

  This directive tells the SSL Module to pick the "best" file locking implementation available to it, choosing between `fcntl` 和 `flock`, in that order. It is only available when the underlying platform and APR supports at least one of the 2.

- `default | yes`

  This directive tells the SSL Module to pick the default locking implementation as determined by the platform and APR.

示例

```
SSLMutex file:/usr/local/apache/logs/ssl_mutex
```

SSLOptions 指令

说明	Configure various SSL engine run-time options
语法	`SSLOptions [+|-]_option_ ...`
作用域	server config, virtual host, directory, .htaccess
覆盖项	Options
状态	扩展(E)
模块	mod_ssl

This directive can be used to control various run-time options on a per-directory basis. Normally, if multiple `SSLOptions` could apply to a directory, then the most specific one is taken completely; the options are not merged. However if *all* the options on the `SSLOptions` directive are preceded by a plus (`+`) or minus (`-`) symbol, the options are merged. Any options preceded by a `+` are added to the options currently in force, and any options preceded by a `-` are removed from the options currently in force.

The available _option_s are:

- `StdEnvVars`

  When this option is enabled, the standard set of SSL related CGI/SSI environment variables are created. This per default is disabled for performance reasons, because the information extraction step is a rather expensive operation. So one usually enables this option for CGI and SSI requests only.

- `CompatEnvVars`

  When this option is enabled, additional CGI/SSI environment variables are created for backward compatibility to other Apache SSL solutions. Look in the Compatibility chapter for details on the particular variables generated.

- `ExportCertData`

  When this option is enabled, additional CGI/SSI environment variables are created: `SSL_SERVER_CERT` , `SSL_CLIENT_CERT` 和 `SSL_CLIENT_CERT_CHAIN_` *n* (with *n* = 0,1,2,..). These contain the PEM-encoded X.509 Certificates of server and client for the current HTTPS connection and can be used by CGI scripts for deeper Certificate checking.

Additionally all other certificates of the client certificate chain are provided, too. This bloats up the environment a little bit which is why you have to use this option to enable it on demand.

- `FakeBasicAuth`

  When this option is enabled, the Subject Distinguished Name (DN) of the Client X509 Certificate is translated into a HTTP Basic Authorization username. This means that the standard Apache authentication methods can be used for access control. The user name is just the Subject of the Client's X509 Certificate (can be determined by running OpenSSL's `openssl x509` command: `openssl x509 -noout -subject -in` *certificate* `.crt`). Note that no password is obtained from the user. Every entry in the user file needs this password: " `xxj31ZMTZzkVA` ", which is the DES-encrypted version of the word " `password` ". Those who live under MD5-based encryption (for instance under FreeBSD or BSD/OS, etc.) should use the following MD5 hash of the same word: " `$1$OXLyS...$Owx8s2/m9/gfkcRVXzgoE/` ".

- `StrictRequire`

  This *forces* forbidden access when `SSLRequireSSL` 或 `SSLRequire` successfully decided that access should be forbidden. Usually the default is that in the case where a " `Satisfy any` " directive is used, and other access restrictions are passed, denial of access due to `SSLRequireSSL` 或 `SSLRequire` is overridden (because that's how the Apache `Satisfy` mechanism should work.) But for strict access restriction you can use `SSLRequireSSL` and/or `SSLRequire` in combination with an " `SSLOptions +StrictRequire` ". Then an additional " `Satisfy Any` " has no chance once mod_ssl has decided to deny access.

- `OptRenegotiate`

  This enables optimized SSL connection renegotiation handling when SSL directives are used in per-directory context. By default a strict scheme is enabled where *every* per-directory reconfiguration of SSL parameters causes a *full* SSL renegotiation handshake. When this option is used mod_ssl tries to avoid unnecessary handshakes by doing more granular (but still safe) parameter checks. Nevertheless these granular checks sometimes maybe not what the user expects, so enable this on a per-directory basis only, please.

示例

```
SSLOptions +FakeBasicAuth -StrictRequire
<Files ~ "\.(cgi|shtml)$">
    SSLOptions +StdEnvVars +CompatEnvVars -ExportCertData
<Files>
```

SSLPassPhraseDialog 指令

说明	Type of pass phrase dialog for encrypted private keys
语法	`SSLPassPhraseDialog _type_`
默认值	`SSLPassPhraseDialog builtin`
作用域	server config
状态	扩展(E)
模块	mod_ssl

When Apache starts up it has to read the various Certificate (see `SSLCertificateFile`) and Private Key (see `SSLCertificateKeyFile`) files of the SSL-enabled virtual servers. Because for security reasons the Private Key files are usually encrypted, mod*ssl needs to query the administrator for a Pass Phrase in order to decrypt those files. This query can be done in two ways which can be configured by _type*:

- `builtin`

  This is the default where an interactive terminal dialog occurs at startup time just before Apache detaches from the terminal. Here the administrator has to manually enter the Pass Phrase for each encrypted Private Key file. Because a lot of SSL-enabled virtual hosts can be configured, the following reuse-scheme is used to minimize the dialog: When a Private Key file is encrypted, all known Pass Phrases (at the beginning there are none, of course) are tried. If one of those known Pass Phrases succeeds no dialog pops up for this particular Private Key file. If none succeeded, another Pass Phrase is queried on the terminal and remembered for the next round (where it perhaps can be reused).

  This scheme allows mod*ssl to be maximally flexible (because for N encrypted Private Key files you _can* use N different Pass Phrases - but then you have to enter all of them, of course) while minimizing the terminal dialog (i.e. when you use a single Pass Phrase for all N Private Key files this Pass Phrase is queried only once).

- `|/path/to/program [args...]`

This mode allows an external program to be used which acts as a pipe to a particular input device; the program is sent the standard prompt text used for the `builtin` mode on `stdin`, and is expected to write password strings on `stdout`. If several passwords are needed (or an incorrect password is entered), additional prompt text will be written subsequent to the first password being returned, and more passwords must then be written back.

- `exec:/path/to/program`

  Here an external program is configured which is called at startup for each encrypted Private Key file. It is called with two arguments (the first is of the form "`servername:portnumber`", the second is either "`RSA`" or "`DSA`"), which indicate for which server and algorithm it has to print the corresponding Pass Phrase to `stdout`. The intent is that this external program first runs security checks to make sure that the system is not compromised by an attacker, and only when these checks were passed successfully it provides the Pass Phrase.

  Both these security checks, and the way the Pass Phrase is determined, can be as complex as you like. Mod_ssl just defines the interface: an executable program which provides the Pass Phrase on `stdout`. Nothing more or less! So, if you're really paranoid about security, here is your interface. Anything else has to be left as an exercise to the administrator, because local security requirements are so different.

  The reuse-algorithm above is used here, too. In other words: The external program is called only once per unique Pass Phrase.

示例

```
SSLPassPhraseDialog exec:/usr/local/apache/sbin/pp-filter
```

SSLProtocol 指令

说明	Configure usable SSL protocol flavors
语法	`SSLProtocol [+|-]_protocol_ ...`
默认值	`SSLProtocol all`
作用域	server config, virtual host
覆盖项	Options
状态	扩展(E)
模块	mod_ssl

This directive can be used to control the SSL protocol flavors mod_ssl should use when establishing its server environment. Clients then can only connect with one of the provided protocols.

The available (case-insensitive) _protocol_s are:

- `SSLv2`

  This is the Secure Sockets Layer (SSL) protocol, version 2.0. It is the original SSL protocol as designed by Netscape Corporation.

- `SSLv3`

  This is the Secure Sockets Layer (SSL) protocol, version 3.0. It is the successor to SSLv2 and the currently (as of February 1999) de-facto standardized SSL protocol from Netscape Corporation. It's supported by almost all popular browsers.

- `TLSv1`

  This is the Transport Layer Security (TLS) protocol, version 1.0. It is the successor to SSLv3 and currently (as of February 1999) still under construction by the Internet Engineering Task Force (IETF). It's still not supported by any popular browsers.

- `All`

  This is a shortcut for " `+SSLv2 +SSLv3 +TLSv1` " and a convinient way for enabling all protocols except one when used in combination with the minus sign on a protocol as the example above shows.

示例

```
#   enable SSLv3 and TLSv1, but not SSLv2

SSLProtocol all -SSLv2
```

SSLProxyCACertificateFile 指令

说明	File of concatenated PEM-encoded CA Certificates for Remote Server Auth
语法	`SSLProxyCACertificateFile _file-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificates of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded Certificate files, in order of preference. This can be used alternatively and/or additionally to `SSLProxyCACertificatePath` .

示例

```
SSLProxyCACertificateFile /usr/local/apache2/conf/ssl.crt/ca-bundle-remote-server.crt
```

SSLProxyCACertificatePath 指令

说明	Directory of PEM-encoded CA Certificates for Remote Server Auth
语法	`SSLProxyCACertificatePath _directory-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive sets the directory where you keep the Certificates of Certification Authorities (CAs) whose remote servers you deal with. These are used to verify the remote server certificate on Remote Server Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value* `.N` . And you should always make sure this directory contains the appropriate symbolic links. Use the `Makefile` which comes with mod_ssl to accomplish this task.

示例

```
SSLProxyCACertificatePath /usr/local/apache2/conf/ssl.crt/
```

SSLProxyCARevocationFile 指令

说明	File of concatenated PEM-encoded CA CRLs for Remote Server Auth
语法	`SSLProxyCARevocationFile _file-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to `SSLProxyCARevocationPath` .

示例

```
SSLProxyCARevocationFile /usr/local/apache2/conf/ssl.crl/ca-bundle-remote-server.crl
```

SSLProxyCARevocationPath 指令

说明	Directory of PEM-encoded CA CRLs for Remote Server Auth
语法	`SSLProxyCARevocationPath _directory-path_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive sets the directory where you keep the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) whose remote servers you deal with. These are used to revoke the remote server certificate on Remote Server Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you have not only to place the CRL files there. Additionally you have to create symbolic links named *hash-value* `.rN` . And you should always make sure this directory contains the appropriate symbolic links. Use the `Makefile` which comes with `mod_ssl` to accomplish this task.

示例

```
SSLProxyCARevocationPath /usr/local/apache2/conf/ssl.crl/
```

SSLProxyCipherSuite 指令

说明	Cipher Suite available for negotiation in SSL proxy handshake
语法	`SSLProxyCipherSuite _cipher-spec_`
默认值	`SSLProxyCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP`
作用域	server config, virtual host, directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_ssl

Equivalent to `SSLCipherSuite`, but for the proxy connection. Please refer to `SSLCipherSuite` for additional information.

SSLProxyEngine 指令

说明	SSL Proxy Engine Operation Switch
语法	`SSLProxyEngine on|off`
默认值	`SSLProxyEngine off`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive toggles the usage of the SSL/TLS Protocol Engine for proxy. This is usually used inside a `<VirtualHost>` section to enable SSL/TLS for proxy usage in a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for proxy image both for the main server and all configured virtual hosts.

示例

```
<VirtualHost _default_:443>
SSLProxyEngine on
...
</VirtualHost>
```

SSLProxyMachineCertificateFile 指令

说明	File of concatenated PEM-encoded client certificates and keys to be used by the proxy
语法	`SSLProxyMachineCertificateFile _filename_`
作用域	server config
覆盖项	Not applicable
状态	扩展(E)
模块	mod_ssl

This directive sets the all-in-one file where you keep the certificates and keys used for authentication of the proxy server to remote servers.

This referenced file is simply the concatenation of the various PEM-encoded certificate files, in order of preference. Use this directive alternatively or additionally to `SSLProxyMachineCertificatePath` .

Currently there is no support for encrypted private keys

示例

```
SSLProxyMachineCertificateFile /usr/local/apache2/conf/ssl.crt/proxy.pem
```

SSLProxyMachineCertificatePath 指令

说明	**Directory of PEM-encoded client certificates and keys to be used by the proxy**
语法	`SSLProxyMachineCertificatePath _directory_`
作用域	server config
覆盖项	Not applicable
状态	扩展(E)
模块	mod_ssl

This directive sets the directory where you keep the certificates and keys used for authentication of the proxy server to remote servers.

The files in this directory must be PEM-encoded and are accessed through hash filenames. Additionally, you must create symbolic links named `_hash-value_.N` . And you should always make sure this directory contains the appropriate symbolic links. Use the Makefile which comes with mod_ssl to accomplish this task.

Currently there is no support for encrypted private keys

示例

```
SSLProxyMachineCertificatePath /usr/local/apache2/conf/proxy.crt/
```

SSLProxyProtocol 指令

说明	**Configure usable SSL protocol flavors for proxy usage**
语法	`SSLProxyProtocol [+|-]_protocol_ ...`
默认值	`SSLProxyProtocol all`
作用域	server config, virtual host
覆盖项	Options
状态	扩展(E)
模块	mod_ssl

This directive can be used to control the SSL protocol flavors mod_ssl should use when establishing its server environment for proxy . It will only connect to servers using one of the provided protocols.

Please refer to `SSLProtocol` for additional information.

SSLProxyVerify 指令

说明	Type of remote server Certificate verification
语法	`SSLProxyVerify _level_`
默认值	`SSLProxyVerify none`
作用域	server config, virtual host, directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_ssl

When a proxy is configured to forward requests to a remote SSL server, this directive can be used to configure certificate verification of the remote server. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the remote server authentication process used in the standard SSL handshake when a connection is established by the proxy. In per-directory context it forces a SSL renegotation with the reconfigured remote server verification level after the HTTP request was read but before the HTTP response is sent.

Note that even when certificate verification is enabled, `mod_ssl` does **not** check whether the `commonName` (hostname) attribute of the server certificate matches the hostname used to connect to the server. In other words, the proxy does not guarantee that the SSL connection to the backend server is "secure" beyond the fact that the certificate is signed by one of the CAs configured using the `SSLProxyCACertificatePath` and/or `SSLProxyCACertificateFile` directives.

The following levels are available for *level*:

- **none**: no remote server Certificate is required at all
- **optional**: the remote server *may* present a valid Certificate
- **require**: the remote server *has to* present a valid Certificate
- **optional_no_ca**: the remote server may present a valid Certificate but it need not to be (successfully) verifiable.

In practice only levels **none**和**require** are really interesting, because level **optional** doesn't work with all servers and level **optional_no_ca** is actually against the idea of authentication (but can be used to establish SSL test pages, etc.)

示例

```
SSLProxyVerify require
```

SSLProxyVerifyDepth 指令

说明	Maximum depth of CA Certificates in Remote Server Certificate verification
语法	`SSLProxyVerifyDepth _number_`
默认值	`SSLProxyVerifyDepth 1`
作用域	server config, virtual host, directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_ssl

This directive sets how deeply mod_ssl should verify before deciding that the remote server does not have a valid certificate. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotation with the reconfigured remote server verification depth after the HTTP request was read but before the HTTP response is sent.

The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are max allowed to be followed while verifying the remote server certificate. A depth of 0 means that self-signed remote server certificates are accepted only, the default depth of 1 means the remote server certificate can be self-signed or has to be signed by a CA which is directly known to the server (i.e. the CA's certificate is under `SSLProxyCACertificatePath`), etc.

示例

```
SSLProxyVerifyDepth 10
```

SSLRandomSeed 指令

说明	Pseudo Random Number Generator (PRNG) seeding source
语法	`SSLRandomSeed _context_ _source_ [_bytes_]`
作用域	server config
状态	扩展(E)
模块	mod_ssl

This configures one or more sources for seeding the Pseudo Random Number Generator (PRNG) in OpenSSL at startup time (*context* is `startup`) and/or just before a new SSL connection is established (*context* is `connect`). This directive can only be used in the global server context because the PRNG is a global facility.

The following *source* variants are available:

- `builtin`

  This is the always available builtin seeding source. It's usage consumes minimum CPU cycles under runtime and hence can be always used without drawbacks. The source used for seeding the PRNG contains of the current time, the current process id and (when applicable) a randomly choosen 1KB extract of the inter-process scoreboard structure of Apache. The drawback is that this is not really a strong source and at startup time (where the scoreboard is still not available) this source just produces a few bytes of entropy. So you should always, at least for the startup, use an additional seeding source.

- `file:/path/to/source`

  This variant uses an external file `/path/to/source` as the source for seeding the PRNG. When *bytes* is specified, only the first *bytes* number of bytes of the file form the entropy (and *bytes* is given to `/path/to/source` as the first argument). When *bytes* is not specified the whole file forms the entropy (and `0` is given to `/path/to/source` as the first argument). Use this especially at startup time, for instance with an available `/dev/random` and/or `/dev/urandom` devices (which usually exist on modern Unix derivates like FreeBSD and Linux).

  But be careful: Usually `/dev/random` provides only as much entropy data as it actually has, i.e. when you request 512 bytes of entropy, but the device currently has only 100 bytes available two things can happen: On some platforms you receive only the 100 bytes while on other platforms the read blocks until enough bytes are available (which can take a long time). Here using an existing `/dev/urandom` is better, because it never blocks and actually gives the amount of requested data. The drawback is just that the quality of the received data may not be the best.

On some platforms like FreeBSD one can even control how the entropy is actually generated, i.e. by which system interrupts. More details one can find under *rndcontrol(8)* on those platforms. Alternatively, when your system lacks such a random device, you can use tool like EGD (Entropy Gathering Daemon) and run it's client program with the `exec:/path/to/program/` variant (see below) or use `egd:/path/to/egd-socket` (see below).

- `exec:/path/to/program`

  This variant uses an external executable `/path/to/program` as the source for seeding the PRNG. When *bytes* is specified, only the first *bytes* number of bytes of its `stdout` contents form the entropy. When *bytes* is not specified, the entirety of the data produced on `stdout` form the entropy. Use this only at startup time when you need a very strong seeding with the help of an external program (for instance as in the example above with the `truerand` utility you can find in the mod*ssl distribution which is based on the AT&T _truerand* library). Using this in the connection context slows down the server too dramatically, of course. So usually you should avoid using external programs in that context.

- `egd:/path/to/egd-socket` (Unix only)

  This variant uses the Unix domain socket of the external Entropy Gathering Daemon (EGD) (see http://www.lothar.com/tech /crypto/) to seed the PRNG. Use this if no random device exists on your platform.

示例

```
SSLRandomSeed startup builtin
SSLRandomSeed startup file:/dev/random
SSLRandomSeed startup file:/dev/urandom 1024
SSLRandomSeed startup exec:/usr/local/bin/truerand 16
SSLRandomSeed connect builtin
SSLRandomSeed connect file:/dev/random
SSLRandomSeed connect file:/dev/urandom 1024
```

SSLRequire 指令

说明	Allow access only when an arbitrarily complex boolean expression is true
语法	`SSLRequire _expression_`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_ssl

This directive specifies a general access requirement which has to be fulfilled in order to allow access. It is a very powerful directive because the requirement specification is an arbitrarily complex boolean expression containing any number of access checks.

The implementation of `SSLRequire` is not thread safe. Using `SSLRequire` inside `.htaccess` files on a threaded MPM may cause random crashes.

The *expression* must match the following syntax (given as a BNF grammar notation):

```
expr     ::= "**true**" | "**false**"
           | "**!**" expr
           | expr "**&&**" expr
           | expr "**||**" expr
           | "**(**" expr "**)**"
           | comp

comp     ::= word "**==**" word | word "**eq**" word
           | word "**!=**" word | word "**ne**" word
           | word "**&lt;**"  word | word "**lt**" word
           | word "**&lt;=**" word | word "**le**" word
           | word "**&gt;**"  word | word "**gt**" word
           | word "**&gt;=**" word | word "**ge**" word
           | word "**in**" "**{**" wordlist "**}**"
           | word "**in**" "**OID(**" word "**)**"
           | word "**=~**" regex
           | word "**!~**" regex

wordlist ::= word
           | wordlist "**,** " word

word     ::= digit
           | cstring
           | variable
           | function

digit    ::= [0-9]+
cstring  ::= "..."
variable ::= "**%{**" varname "**}**"
function ::= funcname "**(**" funcargs "**)**"
```

while for `varname` any variable from Table 3 can be used. Finally for `funcname` the following functions are available:

- `file(` *filename* `)`

This function takes one string argument and expands to the contents of the file. This is especially useful for matching this contents against a regular expression, etc.

Notice that *expression* is first parsed into an internal machine representation and then evaluated in a second step. Actually, in Global and Per-Server Class context *expression* is parsed at startup time and at runtime only the machine representation is executed. For Per-Directory context this is different: here *expression* has to be parsed and immediately executed for every request.

示例

```
SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
            and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
            and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
            and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
            and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20       ) \
           or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
```

`OID()` function expects to find zero or more instances of the given OID in the client certificate, and compares the left-hand side string against the value of matching OID attributes. Every matching OID is checked, until a match is found.

Standard CGI/1.0 and Apache variables:

```
HTTP_USER_AGENT        PATH_INFO             AUTH_TYPE
HTTP_REFERER           QUERY_STRING          SERVER_SOFTWARE
HTTP_COOKIE            REMOTE_HOST           API_VERSION
HTTP_FORWARDED         REMOTE_IDENT          TIME_YEAR
HTTP_HOST              IS_SUBREQ             TIME_MON
HTTP_PROXY_CONNECTION  DOCUMENT_ROOT         TIME_DAY
HTTP_ACCEPT            SERVER_ADMIN          TIME_HOUR
HTTP:headername        SERVER_NAME           TIME_MIN
THE_REQUEST            SERVER_PORT           TIME_SEC
REQUEST_METHOD         SERVER_PROTOCOL       TIME_WDAY
REQUEST_SCHEME         REMOTE_ADDR           TIME
REQUEST_URI            REMOTE_USER           ENV:**variablename**
REQUEST_FILENAME
```

SSL-related variables:

```
HTTPS                    SSL_CLIENT_M_VERSION   SSL_SERVER_M_VERSION
                         SSL_CLIENT_M_SERIAL    SSL_SERVER_M_SERIAL
SSL_PROTOCOL             SSL_CLIENT_V_START     SSL_SERVER_V_START
SSL_SESSION_ID           SSL_CLIENT_V_END       SSL_SERVER_V_END
SSL_CIPHER               SSL_CLIENT_S_DN        SSL_SERVER_S_DN
SSL_CIPHER_EXPORT        SSL_CLIENT_S_DN_C      SSL_SERVER_S_DN_C
SSL_CIPHER_ALGKEYSIZE    SSL_CLIENT_S_DN_ST     SSL_SERVER_S_DN_ST
SSL_CIPHER_USEKEYSIZE    SSL_CLIENT_S_DN_L      SSL_SERVER_S_DN_L
SSL_VERSION_LIBRARY      SSL_CLIENT_S_DN_O      SSL_SERVER_S_DN_O
SSL_VERSION_INTERFACE    SSL_CLIENT_S_DN_OU     SSL_SERVER_S_DN_OU
                         SSL_CLIENT_S_DN_CN     SSL_SERVER_S_DN_CN
                         SSL_CLIENT_S_DN_T      SSL_SERVER_S_DN_T
                         SSL_CLIENT_S_DN_I      SSL_SERVER_S_DN_I
                         SSL_CLIENT_S_DN_G      SSL_SERVER_S_DN_G
                         SSL_CLIENT_S_DN_S      SSL_SERVER_S_DN_S
                         SSL_CLIENT_S_DN_D      SSL_SERVER_S_DN_D
                         SSL_CLIENT_S_DN_UID    SSL_SERVER_S_DN_UID
                         SSL_CLIENT_S_DN_Email  SSL_SERVER_S_DN_Email
                         SSL_CLIENT_I_DN        SSL_SERVER_I_DN
                         SSL_CLIENT_I_DN_C      SSL_SERVER_I_DN_C
                         SSL_CLIENT_I_DN_ST     SSL_SERVER_I_DN_ST
                         SSL_CLIENT_I_DN_L      SSL_SERVER_I_DN_L
                         SSL_CLIENT_I_DN_O      SSL_SERVER_I_DN_O
                         SSL_CLIENT_I_DN_OU     SSL_SERVER_I_DN_OU
                         SSL_CLIENT_I_DN_CN     SSL_SERVER_I_DN_CN
                         SSL_CLIENT_I_DN_T      SSL_SERVER_I_DN_T
                         SSL_CLIENT_I_DN_I      SSL_SERVER_I_DN_I
                         SSL_CLIENT_I_DN_G      SSL_SERVER_I_DN_G
                         SSL_CLIENT_I_DN_S      SSL_SERVER_I_DN_S
                         SSL_CLIENT_I_DN_D      SSL_SERVER_I_DN_D
                         SSL_CLIENT_I_DN_UID    SSL_SERVER_I_DN_UID
                         SSL_CLIENT_I_DN_Email  SSL_SERVER_I_DN_Email
                         SSL_CLIENT_A_SIG       SSL_SERVER_A_SIG
                         SSL_CLIENT_A_KEY       SSL_SERVER_A_KEY
                         SSL_CLIENT_CERT        SSL_SERVER_CERT
                         SSL_CLIENT_CERT_CHAIN_**n**
                         SSL_CLIENT_VERIFY
```

SSLRequireSSL 指令

说明	Deny access when SSL is not used for the HTTP request
语法	`SSLRequireSSL`
作用域	directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_ssl

This directive forbids access unless HTTP over SSL (i.e. HTTPS) is enabled for the current connection. This is very handy inside the SSL-enabled virtual host or directories for defending against configuration errors that expose stuff that should be protected. When this directive is present all requests are denied which are not using SSL.

示例

```
SSLRequireSSL
```

SSLSessionCache 指令

说明	Type of the global/inter-process SSL Session Cache
语法	`SSLSessionCache _type_`
默认值	`SSLSessionCache none`
作用域	server config
状态	扩展(E)
模块	mod_ssl

This configures the storage type of the global/inter-process SSL Session Cache. This cache is an optional facility which speeds up parallel request processing. For requests to the same server process (via HTTP keep-alive), OpenSSL already caches the SSL session information locally. But because modern clients request inlined images and other data via parallel requests (usually up to four parallel requests are common) those requests are served by *different* pre-forked server processes. Here an inter-process cache helps to avoid unneccessary session handshakes.

The following four storage _type_s are currently supported:

- `none`

  This disables the global/inter-process Session Cache. This will incur a noticeable speed penalty and may cause problems if using certain browsers, particularly if client certificates are enabled. This setting is not recommended.

- `nonenotnull`

  This disables any global/inter-process Session Cache. However it does force OpenSSL to send a non-null session ID to accommodate buggy clients that require one.

- `dbm:/path/to/datafile`

  This makes use of a DBM hashfile on the local disk to synchronize the local OpenSSL memory caches of the server processes. This session cache may suffer reliability issues under high load.

- `shm:/path/to/datafile` [`(` *size* `)`]

This makes use of a high-performance cyclic buffer (approx. *size* bytes in size) inside a shared memory segment in RAM (established via `/path/to/datafile`) to synchronize the local OpenSSL memory caches of the server processes. This is the recommended session cache.

- `dc:UNIX:/path/to/socket`

  This makes use of the distcache distributed session caching libraries. The argument should specify the location of the server or proxy to be used using the distcache address syntax; for example, `UNIX:/path/to/socket` specifies a UNIX domain socket (typically a local dc_client proxy); `IP:server.example.com:9001` specifies an IP address.

例子

```
SSLSessionCache dbm:/usr/local/apache/logs/ssl_gcache_data

SSLSessionCache shm:/usr/local/apache/logs/ssl_gcache_data(512000)
```

SSLSessionCacheTimeout 指令

说明	**Number of seconds before an SSL session expires in the Session Cache**
语法	`SSLSessionCacheTimeout _seconds_`
默认值	`SSLSessionCacheTimeout 300`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_ssl

This directive sets the timeout in seconds for the information stored in the global/inter-process SSL Session Cache and the OpenSSL internal memory cache. It can be set as low as 15 for testing, but should be set to higher values like 300 in real life.

示例

```
SSLSessionCacheTimeout 600
```

SSLUserName 指令

说明	Variable name to determine user name
语法	`SSLUserName _varname_`
作用域	server config, directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_ssl
兼容性	仅在 Apache 2.0.51 及以后的版本中可用

This directive sets the "user" field in the Apache request object. This is used by lower modules to identify the user with a character string. In particular, this may cause the environment variable `REMOTE_USER` to be set. The *varname* can be any of the SSL environment variables.

Note that this directive has no effect if the `FakeBasic` option is used (see SSLOptions).

示例

```
SSLUserName SSL_CLIENT_S_DN_CN
```

SSLVerifyClient 指令

说明	Type of Client Certificate verification
语法	`SSLVerifyClient _level_`
默认值	`SSLVerifyClient none`
作用域	server config, virtual host, directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_ssl

This directive sets the Certificate verification level for the Client Authentication. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotation with the reconfigured client verification level after the HTTP request was read but before the HTTP response is sent.

The following levels are available for *level*:

- **none**: no client Certificate is required at all
- **optional**: the client *may* present a valid Certificate
- **require**: the client *has to* present a valid Certificate
- **optional_no_ca**: the client may present a valid Certificate but it need not to be (successfully) verifiable.

In practice only levels **none**和**require** are really interesting, because level **optional** doesn't work with all browsers and level **optional_no_ca** is actually against the idea of authentication (but can be used to establish SSL test pages, etc.)

示例

```
SSLVerifyClient require
```

SSLVerifyDepth 指令

说明	Maximum depth of CA Certificates in Client Certificate verification
语法	`SSLVerifyDepth _number_`
默认值	`SSLVerifyDepth 1`
作用域	server config, virtual host, directory, .htaccess
覆盖项	AuthConfig
状态	扩展(E)
模块	mod_ssl

This directive sets how deeply mod_ssl should verify before deciding that the clients don't have a valid certificate. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotation with the reconfigured client verification depth after the HTTP request was read but before the HTTP response is sent.

The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are max allowed to be followed while verifying the client certificate. A depth of 0 means that self-signed client certificates are accepted only, the default depth of 1 means the client certificate can be self-signed or has to be signed by a CA which is directly known to the server (i.e. the CA's certificate is under `SSLCACertificatePath`), etc.

示例

```
SSLVerifyDepth 10
```

Apache模块 mod_status

说明	生成描述服务器状态的Web页面
状态	基本(B)
模块名	status_module
源文件	mod_status.c

概述

The Status module allows a server administrator to find out how well their server is performing. A HTML page is presented that gives the current server statistics in an easily readable form. If required this page can be made to automatically refresh (given a compatible browser). Another page gives a simple machine-readable list of the current server state.

The details given are:

- The number of worker serving requests
- The number of idle worker
- The status of each worker, the number of requests that worker has performed and the total number of bytes served by the worker (*)
- A total number of accesses and byte count served (*)
- The time the server was started/restarted and the time it has been running for
- Averages giving the number of requests per second, the number of bytes served per second and the average number of bytes per request (*)
- The current percentage CPU used by each worker and in total by Apache (*)
- The current hosts and requests being processed (*)

A compile-time option must be used to display the details marked "(*)" as the instrumentation required for obtaining these statistics does not exist within standard Apache.

Enabling Status Support

To enable status reports only for browsers from the foo.com domain add this code to your `httpd.conf` configuration file

```
<Location /server-status>
SetHandler server-status
Order Deny,Allow
Deny from all
Allow from .foo.com
</Location>
```

You can now access server statistics by using a Web browser to access the page `http://your.server.name/server-status`

Automatic Updates

You can get the status page to update itself automatically if you have a browser that supports "refresh". Access the page `http://your.server.name/server-status?refresh=N` to refresh the page every N seconds.

Machine Readable Status File

A machine-readable version of the status file is available by accessing the page `http://your.server.name/server-status?auto`. This is useful when automatically run, see the Perl program in the `/support` directory of Apache, `log_server_status`.

It should be noted that if `mod_status` is compiled into the server, its handler capability is available in *all* configuration files, including *per*-directory files (例如， `.htaccess`). This may have security-related ramifications for your site.

ExtendedStatus 指令

说明	Keep track of extended status information for each request
语法	`ExtendedStatus On|Off`
默认值	`ExtendedStatus Off`
作用域	server config
状态	基本(B)
模块	mod_status
兼容性	ExtendedStatus is only available in Apache 1.3.2 及以后的版本中可用

This setting applies to the entire server, and cannot be enabled or disabled on a virtualhost-by-virtualhost basis. The collection of extended status information can slow down the server.

Apache模块 mod_suexec

说明	使用与调用web服务器的用户不同的用户身份来运行CGI和SSI程序
状态	扩展(E)
模块名	suexec_module
源文件	mod_suexec.c
兼容性	仅在 Apache 2.0 及以后的版本中可用

概述

将该模块联合 `suexec` 支持程序使用，可以允许以特定的用户和组身份运行CGI程序。

SuexecUserGroup 指令

说明	运行CGI程序的用户和组
语法	`SuexecUserGroup _User Group_`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_suexec
兼容性	仅在 Apache 2.0 及以后的版本中可用

`SuexecUserGroup` 指定CGI程序运行时所使用的用户和组。非CGI程序的请求仍然使用User指令所指定的用户身份处理。该指令取代了Apache1.3的VirtualHosts配置中的User和Group指令。

示例

```
SuexecUserGroup nobody nogroup
```

Apache模块 mod_unique_id

说明	为每个请求生成唯一的标识以便跟踪
状态	扩展(E)
模块名	unique_id_module
源文件	mod_unique_id.c

概述

This module provides a magic token for each request which is guaranteed to be unique across "all" requests under very specific conditions. The unique identifier is even unique across multiple machines in a properly configured cluster of machines. The environment variable `UNIQUE_ID` is set to the identifier for each request. Unique identifiers are useful for various reasons which are beyond the scope of this document.

Theory

First a brief recap of how the Apache server works on Unix machines. This feature currently isn't supported on Windows NT. On Unix machines, Apache creates several children, the children process requests one at a time. Each child can serve multiple requests in its lifetime. For the purpose of this discussion, the children don't share any data with each other. We'll refer to the children as <dfn class="calibre27">httpd processes</dfn>.

Your website has one or more machines under your administrative control, together we'll call them a cluster of machines. Each machine can possibly run multiple instances of Apache. All of these collectively are considered "the universe", and with certain assumptions we'll show that in this universe we can generate unique identifiers for each request, without extensive communication between machines in the cluster.

The machines in your cluster should satisfy these requirements. (Even if you have only one machine you should synchronize its clock with NTP.)

- The machines' times are synchronized via NTP or other network time protocol.
- The machines' hostnames all differ, such that the module can do a hostname lookup on the hostname and receive a different IP address for each machine in the cluster.

As far as operating system assumptions go, we assume that pids (process ids) fit in 32-bits. If the operating system uses more than 32-bits for a pid, the fix is trivial but must be performed in the code.

Given those assumptions, at a single point in time we can identify any httpd process on any machine in the cluster from all other httpd processes. The machine's IP address and the pid of the httpd process are sufficient to do this. So in order to generate unique identifiers for requests we need only distinguish between different points in time.

To distinguish time we will use a Unix timestamp (seconds since January 1, 1970 UTC), and a 16-bit counter. The timestamp has only one second granularity, so the counter is used to represent up to 65536 values during a single second. The quadruple *(ip_addr, pid, time_stamp, counter)* is sufficient to enumerate 65536 requests per second per httpd process. There are issues however with pid reuse over time, and the counter is used to alleviate this issue.

When an httpd child is created, the counter is initialized with (current microseconds divided by 10) modulo 65536 (this formula was chosen to eliminate some variance problems with the low order bits of the microsecond timers on some systems). When a unique identifier is generated, the time stamp used is the time the request arrived at the web server. The counter is incremented every time an identifier is generated (and allowed to roll over).

The kernel generates a pid for each process as it forks the process, and pids are allowed to roll over (they're 16-bits on many Unixes, but newer systems have expanded to 32-bits). So over time the same pid will be reused. However unless it is reused within the same second, it does not destroy the uniqueness of our quadruple. That is, we assume the system does not spawn 65536 processes in a one second interval (it may even be 32768 processes on some Unixes, but even this isn't likely to happen).

Suppose that time repeats itself for some reason. That is, suppose that the system's clock is screwed up and it revisits a past time (or it is too far forward, is reset correctly, and then revisits the future time). In this case we can easily show that we can get pid and time stamp reuse. The choice of initializer for the counter is intended to help defeat this. Note that we really want a random number to initialize the counter, but there aren't any readily available numbers on most systems (*i.e.*, you can't use rand() because you need to seed the generator, and can't seed it with the time because time, at least at one second resolution, has repeated itself). This is not a perfect defense.

How good a defense is it? Suppose that one of your machines serves at most 500 requests per second (which is a very reasonable upper bound at this writing, because systems generally do more than just shovel out static files). To do that it will require a number of children which depends on how many concurrent clients you have. But we'll be pessimistic and suppose that a single child is able to serve 500 requests per second. There are 1000 possible starting counter values such that two sequences of 500 requests overlap. So there is a 1.5% chance that if time (at one second resolution) repeats itself this child will repeat a

counter value, and uniqueness will be broken. This was a very pessimistic example, and with real world values it's even less likely to occur. If your system is such that it's still likely to occur, then perhaps you should make the counter 32 bits (by editing the code).

You may be concerned about the clock being "set back" during summer daylight savings. However this isn't an issue because the times used here are UTC, which "always" go forward. Note that x86 based Unixes may need proper configuration for this to be true -- they should be configured to assume that the motherboard clock is on UTC and compensate appropriately. But even still, if you're running NTP then your UTC time will be correct very shortly after reboot.

`UNIQUE_ID` environment variable is constructed by encoding the 112-bit (32-bit IP address, 32 bit pid, 32 bit time stamp, 16 bit counter) quadruple using the alphabet `[A-Za-z0-9@-]` in a manner similar to MIME base64 encoding, producing 19 characters. The MIME base64 alphabet is actually `[A-Za-z0-9+/]` however `+` 和 `/` need to be specially encoded in URLs, which makes them less desirable. All values are encoded in network byte ordering so that the encoding is comparable across architectures of different byte ordering. The actual ordering of the encoding is: time stamp, IP address, pid, counter. This ordering has a purpose, but it should be emphasized that applications should not dissect the encoding. Applications should treat the entire encoded `UNIQUE_ID` as an opaque token, which can be compared against other `UNIQUE_ID`s for equality only.

The ordering was chosen such that it's possible to change the encoding in the future without worrying about collision with an existing database of `UNIQUE_ID`s. The new encodings should also keep the time stamp as the first element, and can otherwise use the same alphabet and bit length. Since the time stamps are essentially an increasing sequence, it's sufficient to have a *flag second* in which all machines in the cluster stop serving and request, and stop using the old encoding format. Afterwards they can resume requests and begin issuing the new encodings.

This we believe is a relatively portable solution to this problem. It can be extended to multithreaded systems like Windows NT, and can grow with future needs. The identifiers generated have essentially an infinite life-time because future identifiers can be made longer as required. Essentially no communication is required between machines in the cluster (only NTP synchronization is required, which is low overhead), and no communication between httpd processes is required (the communication is implicit in the pid value assigned by the kernel). In very specific situations the identifier can be shortened, but more information needs to be assumed (for example the 32-bit IP address is overkill for any site, but there is no portable shorter replacement for it).

Apache模块 mod_userdir

说明	允许用户从自己的主目录中提供页面(使用"/~username")
状态	基本(B)
模块名	userdir_module
源文件	mod_userdir.c

概述

此模块允许使用类似 `http://example.com/~user/` 的语法来访问用户网站目录。

UserDir 指令

说明	用户网站目录的位置
语法	`UserDir _directory-filename_`
作用域	server config, virtual host
状态	基本(B)
模块	mod_userdir

`UserDir` 指令指定了用户目录下的一个实实在在的目录，存放了该用户提供访问的文档。*Directory-filename*可以是以下几种形式之一：

- 一个目录名或如下所示的匹配模式：
- 关键词 `disabled` 停止所有用户名到目录的转换，但不包括明确使用 `enabled` 启用的(见下面)目录。
- 关键词 `disabled` 并跟随一个以空格分隔的用户名列表(其中的用户即使出现在 `enabled` 的用户列表中，也不会进行目录转换)。
- 关键词 `enabled` 并跟随一个以空格分隔的用户名列表。此列表中的用户允许进行目录转换，即使有一个全局的 `disabled` 关闭了此操作，但是，如果同时出现在 `disabled` 的用户列表中，则不执行转换操作。

如果在 `Userdir` 指令中，既没有 `enabled` 也没有 `disabled` 关键词，则其参数将被视为文件匹配模式，用于转换成目录名。对 `http://www.foo.com/~bob/one/two.html` 的请求会被转换为：

UserDir 指令	转换后的路径
UserDir public_html	~bob/public_html/one/two.html
UserDir /usr/web	/usr/web/bob/one/two.html
UserDir /home/*/www	/home/bob/www/one/two.html

下列指令将发送重定向到客户端：

UserDir 指令	转换后的路径
UserDir http://www.foo.com/users	http://www.foo.com/users/bob/one/two.html
UserDir http://www.foo.com/*/usr	http://www.foo.com/bob/usr/one/two.html
UserDir http://www.foo.com/~*/	http://www.foo.com/~bob/one/two.html

注意：使用此指令时要很小心，例如"`UserDir ./`"可能会把"`/~root`映射到"`/`"而这可能不是我们想要的。强烈建议在配置文件中包含一个"`UserDir disabled root`"声明。更多信息请参见`Directory`指令和安全提示。

举例：

允许某些用户使用`UserDir`指令，而禁止其他用户：

```
UserDir disabled
UserDir enabled user1 user2 user3
```

允许大多数用户使用`UserDir`指令，而禁止一小部分用户：

```
UserDir enabled
UserDir disabled user4 user5 user6
```

还可以指定任选其一的(alternative)用户网站目录：

```
Userdir public_html /usr/web http://www.foo.com/
```

对http://www.foo.com/~bob/one/two.html的请求，会首先尝试获取"~bob/public_html/one/two.html"，其次是"/usr/web/bob/one/two.html"，最后产生一个到http://www.foo.com/bob/one/two.html的重定向。

如果要增加重定向，则必须放在列表的最后。因为Apache不能判断重定向是否成功，所以如果不放在最后，那么它只是一个替换地址。

2.1.4及以后的版本中，默认不开启用户网站目录。在未设置`UserDir`指令的情况下将使用"`UserDir public_html`"默认值。

参见

- 用户网站目录

Apache模块 mod_usertrack

说明	使用Session跟踪用户(会发送很多Cookie)，以记录用户的点击流
状态	扩展(E)
模块名	usertrack_module
源文件	mod_usertrack.c

概述

Previous releases of Apache have included a module which generates a 'clickstream' log of user activity on a site using cookies. This was called the "cookies" module, mod_cookies. In Apache 1.2 and later this module has been renamed the "user tracking" module, mod_usertrack. This module has been simplified and new directives added.

Logging

Previously, the cookies module (now the user tracking module) did its own logging, using the `CookieLog` directive. In this release, this module does no logging at all. Instead, a configurable log format file should be used to log user click-streams. This is possible because the logging module now allows multiple log files. The cookie itself is logged by using the text `%{cookie}n` in the log file format. For example:

```
CustomLog logs/clickstream "%{cookie}n %r %t"
```

For backward compatibility the configurable log module implements the old `CookieLog` directive, but this should be upgraded to the above `CustomLog` directive.

2-digit or 4-digit dates for cookies?

(the following is from message <022701bda43d$9d32bbb0$1201a8c0@christian.office.sane.com> in the new-httpd archives)

```
From: "Christian Allen" <christian@sane.com>
Subject: Re: Apache Y2K bug in mod_usertrack.c
Date: Tue, 30 Jun 1998 11:41:56 -0400

Did some work with cookies and dug up some info that might be useful.

True, Netscape claims that the correct format NOW is four digit dates, and
four digit dates do in fact work... for Netscape 4.x (Communicator), that
is.  However, 3.x and below do NOT accept them.  It seems that Netscape
originally had a 2-digit standard, and then with all of the Y2K hype and
probably a few complaints, changed to a four digit date for Communicator.
Fortunately, 4.x also understands the 2-digit format, and so the best way to
ensure that your expiration date is legible to the client's browser is to
use 2-digit dates.

However, this does not limit expiration dates to the year 2000; if you use
an expiration year of "13", for example, it is interpreted as 2013, NOT
1913!  In fact, you can use an expiration year of up to "37", and it will be
understood as "2037" by both MSIE and Netscape versions 3.x and up (not sure
about versions previous to those).  Not sure why Netscape used that
particular year as its cut-off point, but my guess is that it was in respect
to UNIX's 2038 problem.  Netscape/MSIE 4.x seem to be able to understand
2-digit years beyond that, at least until "50" for sure (I think they
understand up until about "70", but not for sure).

Summary:  Mozilla 3.x and up understands two digit dates up until "37"
(2037).  Mozilla 4.x understands up until at least "50" (2050) in 2-digit
form, but also understands 4-digit years, which can probably reach up until
9999\.  Your best bet for sending a long-life cookie is to send it for some
time late in the year "37".
```

CookieDomain 指令

说明	The domain to which the tracking cookie applies
语法	`CookieDomain _domain_`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_usertrack

This directive controls the setting of the domain to which the tracking cookie applies. If not present, no domain is included in the cookie header field.

The domain string **must** begin with a dot, and **must** include at least one embedded dot. That is, ".foo.com" is legal, but "foo.bar.com" and ".com" are not.

CookieExpires 指令

说明	Expiry time for the tracking cookie
语法	`CookieExpires _expiry-period_`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_usertrack

When used, this directive sets an expiry time on the cookie generated by the usertrack module. The *expiry-period* can be given either as a number of seconds, or in the format such as "2 weeks 3 days 7 hours". Valid denominations are: years, months, weeks, days, hours, minutes and seconds. If the expiry time is in any format other than one number indicating the number of seconds, it must be enclosed by double quotes.

If this directive is not used, cookies last only for the current browser session.

CookieName 指令

说明	Name of the tracking cookie
语法	`CookieName _token_`
默认值	`CookieName Apache`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_usertrack

This directive allows you to change the name of the cookie this module uses for its tracking purposes. By default the cookie is named " `Apache` ".

You must specify a valid cookie name; results are unpredictable if you use a name containing unusual characters. Valid characters include A-Z, a-z, 0-9, "_", and "-".

CookieStyle 指令

说明	Format of the cookie header field
语法	`CookieStyle _Netscape|Cookie|Cookie2|RFC2109|RFC2965_`
默认值	`CookieStyle Netscape`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_usertrack

This directive controls the format of the cookie header field. The three formats allowed are:

- **Netscape**, which is the original but now deprecated syntax. This is the default, and the syntax Apache has historically used.
- **Cookie或RFC2109**, which is the syntax that superseded the Netscape syntax.
- **Cookie2或RFC2965**, which is the most current cookie syntax.

Not all clients can understand all of these formats. but you should use the newest one that is generally acceptable to your users' browsers.

CookieTracking 指令

说明	Enables tracking cookie
语法	`CookieTracking on|off`
默认值	`CookieTracking off`
作用域	server config, virtual host, directory, .htaccess
覆盖项	FileInfo
状态	扩展(E)
模块	mod_usertrack

When the user track module is compiled in, and "CookieTracking on" is set, Apache will start sending a user-tracking cookie for all new requests. This directive can be used to turn this behavior on or off on a per-server or per-directory basis. By default, compiling mod_usertrack will not activate cookies.

Apache模块 mod_version

说明	提供基于版本的配置段支持
状态	扩展(E)
模块名	version_module
源文件	mod_version.c
兼容性	仅在 Apache 2.0.56 及以后的版本中可用

概述

This module is designed for the use in test suites and large networks which have to deal with different httpd versions and different configurations. It provides a new container -- `<IfVersion>` , which allows a flexible version checking including numeric comparisons and regular expressions.

例子

```
      <IfVersion 2.1.0>

 # current httpd version is exactly 2.1.0
      </IfVersion>

      <IfVersion >= 2.2>

 # use really new features :-)
      </IfVersion>
```

See below for further possibilities.

<IfVersion> 指令

说明	**contains version dependent configuration**
语法	`<IfVersion [[!]operator] version> ... </IfVersion>`
作用域	server config, virtual host, directory, .htaccess
覆盖项	All
状态	扩展(E)
模块	mod_version

<IfVersion> section encloses configuration directives which are executed only if the httpd version matches the desired criteria. For normal (numeric) comparisons the version argument has the format major[.minor[.patch]], e.g. 2.1.0 或 2.2. minor和patch are optional. If these numbers are omitted, they are assumed to be zero. The following numerical operators are possible:

operator	description
= 或 ==	httpd version is equal
>	httpd version is greater than
>=	httpd version is greater or equal
<	httpd version is less than
<=	httpd version is less or equal

示例

```
      <IfVersion >= 2.1>

  # this happens only in versions greater or

         # equal 2.1.0.
      </IfVersion>
```

Besides the numerical comparison it is possible to match a regular expression against the httpd version. There are two ways to write it:

operator	description
= 或 ==	version has the form /regex/
~	version has the form regex

示例

```
      <IfVersion = /^2.1.[01234]$/>

  # e.g. workaround for buggy versions
      </IfVersion>
```

In order to reverse the meaning, all operators can be preceded by an exclamation mark (!):

```
      <IfVersion !~ ^2.1.[01234]$>

  # not for those versions
      </IfVersion>
```

If the operator is omitted, it is assumed to be `=`.

Apache模块 mod_vhost_alias

说明	提供大批量虚拟主机的动态配置支持
状态	扩展(E)
模块名	vhost_alias_module
源文件	mod_vhost_alias.c

概述

本模块通过将HTTP请求中的IP地址和/或" `Host:` "头内容转换为所要提供服务的文件路径名来创建动态的虚拟主机配置。这样的做法，使得应用配置大量相似的虚拟主机变得更为容易。

注意

如果使用 `mod_alias` 或 `mod_userdir` 来将URI转换为文件名，那么 `mod_vhost_alias` 的设定将被覆盖。例如，下面的配置将始终把 `/cgi-bin/script.pl` 映射为 `/usr/local/apache2/cgi-bin/script.pl` 。

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
VirtualScriptAlias /never/found/%0/cgi-bin/
```

目录名称的转换

本模块中的所有指令都用于将字符串替换为路径名。被替换的字符串(以后称为"name")可以是服务器名(参见 `UseCanonicalName` 指令以了解决策方法)或者是"点数字"格式的虚拟主机IP地址。替换操作由 `printf` 格式修饰符控制，该修饰符有以下几种格式：

`%%`	插入一个百分号(`%`)
`%p`	插入虚拟主机的端口号
`%N.M`	插入名称(或者名称的一部分)

`N` 和 `M` 被用来指定name中的子字符串。 `N` 从name中用小数点分隔的某部分中选取，而 `M` 是从 `N` 选中的字符串中选取部分字符。 `M` 是可选的且默认为"0"；小数点当且仅当 `M` 存在时才必须书写。替换操作如下：

`0`	整个name
`1`	第一部分
`2`	第二部分
`-1`	最后一部分
`-2`	倒数第二部分
`2+`	从第二部分开始到最后的所有部分
`-2+`	倒数第二部分以及之前的各部分
`1+` 和 `-1+`	等同于 `0`

如果 `N` 或 `M` 大于部分的个数，则简单的用下划线来替换。

示例

对于一个简单的基于名称的虚拟主机，配置文件中可能会使用下面的指令：

```
UseCanonicalName    Off
VirtualDocumentRoot /usr/local/apache/vhosts/%0
```

那么对 `http://www.example.com/directory/file.html` 的请求将会返回文件 `/usr/local/apache/vhosts/www.example.com/directory/file.html`

对于拥有大量虚拟主机的情况而言，减少 `vhosts` 目录大小的一个好办法就是重新组织。为此你可以使用下面的配置：

```
UseCanonicalName    Off
VirtualDocumentRoot /usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2
```

那么对 `http://www.domain.example.com/directory/file.html` 的请求将会返回文件 `/usr/local/apache/vhosts/example.com/d/o/m/domain/directory/file.html`

进一步的分割可以用name尾字符来索引(hashing)，例如：

```
VirtualDocumentRoot /usr/local/apache/vhosts/%3+/%2.-1/%2.-2/%2.-3/%2
```

该例返回文件 `/usr/local/apache/vhosts/example.com/n/i/a/domain/directory/file.html`

也可以这样使用：

```
VirtualDocumentRoot /usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2.4+
```

该例返回文件 `/usr/local/apache/vhosts/example.com/d/o/m/ain/directory/file.html`

对于基于IP地址的虚拟主机，可以这样配置：

```
UseCanonicalName DNS
VirtualDocumentRootIP /usr/local/apache/vhosts/%1/%2/%3/%4/docs
VirtualScriptAliasIP  /usr/local/apache/vhosts/%1/%2/%3/%4/cgi-bin
```

对 `http://www.domain.example.com/directory/file.html` 的请求将会返回文件 `/usr/local/apache/vhosts/10/20/30/40/docs/directory/file.html` ，这里假设 `www.domain.example.com` 的IP地址为10.20.30.40。
对 `http://www.domain.example.com/cgi-bin/script.pl` 的请求将会执行程序 `/usr/local/apache/vhosts/10/20/30/40/cgi-bin/script.pl`

如果你希望在 `VirtualDocumentRoot` 指令中包含点字符(`.`)，但这又和 `%` 指令产生冲突，可以这样解决：

```
VirtualDocumentRoot /usr/local/apache/vhosts/%2.0.%3.0
```

对 `http://www.domain.example.com/directory/file.html` 的请求将会返回文件 `/usr/local/apache/vhosts/domain.example/directory/file.html`

`LogFormat` 指令的 `%V` 和 `%A` 在和本模块的协同中起了一定作用。

VirtualDocumentRoot 指令

说明	对于给定的基于名称的虚拟主机动态配置根文档目录
语法	`VirtualDocumentRoot _interpolated-directory_|none`
默认值	`VirtualDocumentRoot none`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_vhost_alias

`VirtualDocumentRoot` 指令使Apache可以通过虚拟主机的域名找到相应的文档。扩展*interpolated-directory*所得到的目录将会作为虚拟主机的根目录，这和 `DocumentRoot` 指令的参数是一样的。如果*interpolated-directory*为 `none` ，那么 `VirtualDocumentRoot` 将被关闭。该指令不能和 `VirtualDocumentRootIP` 指令在同一作用域中使用。

VirtualDocumentRootIP 指令

说明	对于给定的基于IP的虚拟主机动态配置根文档目录
语法	`VirtualDocumentRootIP _interpolated-directory_|none`
默认值	`VirtualDocumentRootIP none`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_vhost_alias

`VirtualDocumentRootIP` 与 `VirtualDocumentRoot` 相似，只是替换操作时用的不是虚拟主机名称，而是IP地址。

VirtualScriptAlias 指令

说明	对于给定的基于名称的虚拟主机动态配置CGI目录
语法	`VirtualScriptAlias _interpolated-directory_|none`
默认值	`VirtualScriptAlias none`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_vhost_alias

`VirtualScriptAlias` 指令使Apache确定何处存放的是CGI脚本，这和 `VirtualDocumentRoot` 的做法是一样的。它匹配请求中的以 `/cgi-bin/` 开始的URI，更像" `ScriptAlias` `/cgi-bin/` "的作用。

VirtualScriptAliasIP 指令

说明	对于给定的基于IP的虚拟主机动态配置CGI目录
语法	`VirtualScriptAliasIP _interpolated-directory_|none`
默认值	`VirtualScriptAliasIP none`
作用域	server config, virtual host
状态	扩展(E)
模块	mod_vhost_alias

`VirtualScriptAliasIP` 和 `VirtualScriptAlias` 相似，只是替换操作使用的不是虚拟主机名称，而是IP地址。

``

Developer Documentation for Apache 2.0

Many of the documents on these Developer pages are lifted from Apache 1.3's documentation. While they are all being updated to Apache 2.0, they are in different stages of progress. Please be patient, and point out any discrepancies or errors on the developer/ pages directly to the dev@httpd.apache.org mailing list.

Topics

- Apache 1.3 API Notes
- Apache 2.0 Hook Functions
- Request Processing in Apache 2.0
- How filters work in Apache 2.0
- Converting Modules from Apache 1.3 to Apache 2.0
- Debugging Memory Allocation in APR
- Documenting Apache 2.0
- Apache 2.0 Thread Safety Issues

External Resources

- Tools provided by Ian Holsman:
  - Apache 2 cross reference
  - Autogenerated Apache 2 code documentation
- Module Development Tutorials by Kevin O'Donnell
  - Integrating a module into the Apache build system
  - Handling configuration directives
- Some notes on Apache module development by Ryan Bloom
- Developer articles at apachetutor include:
  - Request Processing in Apache
  - Configuration for Modules
  - Resource Management in Apache
  - Connection Pooling in Apache
  - Introduction to Buckets and Brigades

Apache 1.3 API notes

Warning

This document has not been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

These are some notes on the Apache API and the data structures you have to deal with, *etc.* They are not yet nearly complete, but hopefully, they will help you get your bearings. Keep in mind that the API is still subject to change as we gain experience with it. (See the TODO file for what *might* be coming). However, it will be easy to adapt modules to any changes that are made. (We have more modules to adapt than you do).

A few notes on general pedagogical style here. In the interest of conciseness, all structure declarations here are incomplete -- the real ones have more slots that I'm not telling you about. For the most part, these are reserved to one component of the server core or another, and should be altered by modules with caution. However, in some cases, they really are things I just haven't gotten around to yet. Welcome to the bleeding edge.

Finally, here's an outline, to give you some bare idea of what's coming up, and in what order:

``

Basic concepts

We begin with an overview of the basic concepts behind the API, and how they are manifested in the code.

Handlers, Modules, and Requests

Apache breaks down request handling into a series of steps, more or less the same way the Netscape server API does (although this API has a few more stages than NetSite does, as hooks for stuff I thought might be useful in the future). These are:

- URI -> Filename translation
- Auth ID checking [is the user who they say they are?]
- Auth access checking [is the user authorized *here*?]
- Access checking other than auth
- Determining MIME type of the object requested
- 'Fixups' -- there aren't any of these yet, but the phase is intended as a hook for possible extensions like `SetEnv`, which don't really fit well elsewhere.
- Actually sending a response back to the client.
- Logging the request

These phases are handled by looking at each of a succession of *modules*, looking to see if each of them has a handler for the phase, and attempting invoking it if so. The handler can typically do one of three things:

- *Handle* the request, and indicate that it has done so by returning the magic constant `OK`.
- *Decline* to handle the request, by returning the magic integer constant `DECLINED`. In this case, the server behaves in all respects as if the handler simply hadn't been there.
- Signal an error, by returning one of the HTTP error codes. This terminates normal handling of the request, although an ErrorDocument may be invoked to try to mop up, and it will be logged in any case.

Most phases are terminated by the first module that handles them; however, for logging, 'fixups', and non-access authentication checking, all handlers always run (barring an error). Also, the response phase is unique in that modules may declare multiple handlers for it, via a dispatch table keyed on the MIME type of the requested object. Modules may declare a response-phase handler which can handle *any* request, by giving it the key `*/*` (*i.e.*, a wildcard MIME type specification). However, wildcard handlers are only invoked if the server has already tried and failed to find a more specific response handler for the MIME type of the requested object (either none existed, or they all declined).

The handlers themselves are functions of one argument (a `request_rec` structure. vide infra), which returns an integer, as above.

A brief tour of a module

At this point, we need to explain the structure of a module. Our candidate will be one of the messier ones, the CGI module -- this handles both CGI scripts and the `ScriptAlias` config file command. It's actually a great deal more complicated than most modules, but if we're going to have only one example, it might as well be the one with its fingers in every place.

Let's begin with handlers. In order to handle the CGI scripts, the module declares a response handler for them. Because of `ScriptAlias`, it also has handlers for the name translation phase (to recognize `ScriptAlias`ed URIs), the type-checking phase (any `ScriptAlias`ed request is typed as a CGI script).

The module needs to maintain some per (virtual) server information, namely, the `ScriptAlias`es in effect; the module structure therefore contains pointers to a functions which builds these structures, and to another which combines two of them (in case the main server and a virtual server both have `ScriptAlias`es declared).

Finally, this module contains code to handle the `ScriptAlias` command itself. This particular module only declares one command, but there could be more, so modules have *command tables* which declare their commands, and describe where they are permitted, and how they are to be invoked.

A final note on the declared types of the arguments of some of these commands: a `pool` is a pointer to a *resource pool* structure; these are used by the server to keep track of the memory which has been allocated, files opened, *etc.*, either to service a particular request, or to handle the process of configuring itself. That way, when the request is over (or, for the configuration pool, when the server is restarting), the memory can be freed, and the files closed, *en masse*, without anyone having to write explicit code to track them all down and dispose of them. Also, a `cmd_parms` structure contains various information about the config file being read, and other status information, which is sometimes of use to the function which processes a config-file command (such as `ScriptAlias`). With no further ado, the module itself:

`

```
/* Declarations of handlers. */

       int translate_scriptalias (request_rec *);

       int type_scriptalias (request_rec *);

       int cgi_handler (request_rec *);

       /* Subsidiary dispatch table for response-phase

        * handlers, by MIME type */

       handler_rec cgi_handlers[] = {

{ "application/x-httpd-cgi", cgi_handler },

         { NULL }
       };

       /* Declarations of routines to manipulate the

        * module's configuration info.  Note that these are

        * returned, and passed in, as void *'s; the server

        * core keeps track of them, but it doesn't, and can't,

        * know their internal structure.

        */

       void *make_cgi_server_config (pool *);

       void *merge_cgi_server_config (pool *, void *, void *);

       /* Declarations of routines to handle config-file commands */

       extern char *script_alias(cmd_parms *, void *per_dir_config, char *fake,
                                 char *real);

       command_rec cgi_cmds[] = {

{ "ScriptAlias", script_alias, NULL, RSRC_CONF, TAKE2,

"a fakename and a realname"},

         { NULL }
       };

       module cgi_module = {
```

STANDARD_MODULE_STUFF, NULL, / *initializer* / NULL, / *dir config creator* / NULL, / *dir merger* / make_cgi_server_config, / *server config* / merge_cgi_server_config, / *merge server config* / cgi_cmds, / *command table* / cgi_handlers, / *handlers* / translate_scriptalias, / *filename translation* / NULL, / *check_user_id* / NULL, / *check auth* / NULL, / *check access* / type_scriptalias, / *type_checker* / NULL, / *fixups* / NULL, / *logger* / NULL / *header parser* / };

``

How handlers work

The sole argument to handlers is a `request_rec` structure. This structure describes a particular request which has been made to the server, on behalf of a client. In most cases, each connection to the client generates only one `request_rec` structure.

A brief tour of the request_rec

`request_rec` contains pointers to a resource pool which will be cleared when the server is finished handling the request; to structures containing per-server and per-connection information, and most importantly, information on the request itself.

The most important such information is a small set of character strings describing attributes of the object being requested, including its URI, filename, content-type and content-encoding (these being filled in by the translation and type-check handlers which handle the request, respectively).

Other commonly used data items are tables giving the MIME headers on the client's original request, MIME headers to be sent back with the response (which modules can add to at will), and environment variables for any subprocesses which are spawned off in the course of servicing the request. These tables are manipulated using the `ap_table_get` and `ap_table_set` routines.

Note that the `Content-type` header value *cannot* be set by module content-handlers using the `ap_table_*()` routines. Rather, it is set by pointing the `content_type` field in the `request_rec` structure to an appropriate string. 例如，

```
r->content_type = "text/html";
```

Finally, there are pointers to two data structures which, in turn, point to per-module configuration structures. Specifically, these hold pointers to the data structures which the module has built to describe the way it has been configured to operate in a given directory (via `.htaccess` files or `<Directory>` sections), for private data it has built in the course of servicing the request (so modules' handlers for one phase can pass 'notes' to their handlers for other phases). There is another such configuration vector in the `server_rec` data structure pointed to by the `request_rec` , which contains per (virtual) server configuration data.

Here is an abridged declaration, giving the fields most commonly used:

`

```
struct request_rec {
        pool *pool;
        conn_rec *connection;
        server_rec *server;
        /* What object is being requested */
        char *uri;
        char *filename;
        char *path_info;
```

char *args;* / QUERY_ARGS, if any / *struct stat finfo;* / Set by server core;

```
                   * st_mode set to zero if no such file */
```

```
`char *content_type;
        char *content_encoding;
        /* MIME header environments, in and out. Also,
         * an array containing environment variables to
         * be passed to subprocesses, so people can write
         * modules to add to that environment.
         *
         * The difference between headers_out and
         * err_headers_out is that the latter are printed
         * even on error, and persist across internal
         * redirects (so the headers printed for
         *` `ErrorDocument` handlers will have
         them).
         */
        table *headers_in;
        table *headers_out;
        table *err_headers_out;
        table *subprocess_env;
        /* Info about the request itself... */
```

int header*only; / HEAD request, as opposed to GET / char protocol; / Protocol, as given to us, or HTTP/0.9 / char method; /* GET, HEAD, POST, _etc. / int method_number; /* MGET, *M_POST, _etc.* */

```
`/* Info for logging */

        char *the_request;

        int bytes_sent;

        /* A flag which modules can set, to indicate that

         * the data being returned is volatile, and clients

         * should be told not to cache it.

         */

        int no_cache;

        /* Various other config info which may change

         * with .htaccess files

         * These are config vectors, with one void*

         * pointer for each module (the thing pointed

         * to being the module's business).

         */`
```

void *per_dir_config; /* Options set in config files, *etc.* / *void* request_config; / *Notes on* this *request* /

```
 `};`

### `Where request_rec structures come from`

 Most `request_rec` structures are built by reading an HTTP
      request from a client, and filling in the fields. However, there are a
      few exceptions:

*   If the request is to an imagemap, a type map (_i.e._, a
          `*.var` file), or a CGI script which returned a local
          'Location:', then the resource which the user requested is going to be
          ultimately located by some URI other than what the client originally
          supplied. In this case, the server does an _internal redirect_,
          constructing a new `request_rec` for the new URI, and
          processing it almost exactly as if the client had requested the new URI
          directly.

*   If some handler signaled an error, and an `ErrorDocument`
          is in scope, the same internal redirect machinery comes into play.

*   Finally, a handler occasionally needs to investigate 'what would
          happen if' some other request were run. For instance, the directory
          indexing module needs to know what MIME type would be assigned to a
          request for each directory entry, in order to figure out what icon to
          use.

    Such handlers can construct a _sub-request_, using the
          functions `ap_sub_req_lookup_file`,
          `ap_sub_req_lookup_uri`, and `ap_sub_req_method_uri`;
          these construct a new `request_rec` structure and processes it
          as you would expect, up to but not including the point of actually sending
          a response. (These functions skip over the access checks if the
          sub-request is for a file in the same directory as the original
          request).
```

```
    (Server-side includes work by building sub-requests and then actually
          invoking the response handler for them, via the function
          `ap_run_sub_req`).

### &lt;a name="req_return" id="calibre_link-432" class="pcalibre1 calibre22 pcalibre3 pc
    error codes&lt;/a&gt;

As discussed above, each handler, when invoked to handle a particular
      `request_rec`, has to return an `int` to indicate
      what happened. That can either be

*   `OK` -- the request was handled successfully. This may or
          may not terminate the phase.

*   `DECLINED` -- no erroneous condition exists, but the module
          declines to handle the phase; the server tries to find another.

*   an HTTP error code, which aborts handling of the request.

Note that if the error code returned is `REDIRECT`, then
      the module should put a `Location` in the request's
      `headers_out`, to indicate where the client should be
      redirected _to_.

### &lt;a name="resp_handlers" id="calibre_link-433" class="pcalibre1 calibre22 pcalibre3
    handlers&lt;/a&gt;

Handlers for most phases do their work by simply setting a few fields
      in the `request_rec` structure (or, in the case of access
      checkers, simply by returning the correct error code). However, response
      handlers have to actually send a request back to the client.

They should begin by sending an HTTP response header, using the
      function `ap_send_http_header`. (You don't have to do anything
      special to skip sending the header for HTTP/0.9 requests; the function
      figures out on its own that it shouldn't do anything). If the request is
      marked `header_only`, that's all they should do; they should
      return after that, without attempting any further output.

Otherwise, they should produce a request body which responds to the
      client as appropriate. The primitives for this are `ap_rputc`
      and `ap_rprintf`, for internally generated output, and
      `ap_send_fd`, to copy the contents of some `FILE *`
      straight to the client.

At this point, you should more or less understand the following piece
      of code, which is the handler which handles `GET` requests
      which have no more specific handler; it also shows how conditional
      `GET`s can be handled, if it's desirable to do so in a
      particular response handler -- `ap_set_last_modified` checks
      against the `If-modified-since` value supplied by the client,
      if any, and returns an appropriate code (which will, if nonzero, be
      USE_LOCAL_COPY). No similar considerations apply for
      `ap_set_content_length`, but it returns an error code for
      symmetry.
```

```
    int default_handler (request_rec *r)

    {
```

int errstatus;

```
    FILE *f;

    if (r->method_number != M_GET) return DECLINED;

    if (r->finfo.st_mode == 0) return NOT_FOUND;

    if ((errstatus = ap_set_content_length (r, r->finfo.st_size))

        ||
       (errstatus = ap_set_last_modified (r, r->finfo.st_mtime)))

    return errstatus;

    f = fopen (r->filename, "r");

    if (f == NULL) {
```

log_reason("file permissions deny server access", r->filename, r);

```
      return FORBIDDEN;
    }

    register_timeout ("send", r);

    ap_send_http_header (r);

    if (!r->header_only) send_fd (f, r);

    ap_pfclose (r->pool, f);

    return OK;
  }
```

```
Finally, if all of this is too much of a challenge, there are a few
      ways out of it. First off, as shown above, a response handler which has
      not yet produced any output can simply return an error code, in which
      case the server will automatically produce an error response. Secondly,
      it can punt to some other handler by invoking
      `ap_internal_redirect`, which is how the internal redirection
      machinery discussed above is invoked. A response handler which has
      internally redirected should always return `OK`.

(Invoking `ap_internal_redirect` from handlers which are
      _not_ response handlers will lead to serious confusion).

### &lt;a name="auth_handlers" id="calibre_link-434" class="pcalibre1 calibre22 pcalibre3
    handlers&lt;/a&gt;

Stuff that should be discussed here in detail:

*   Authentication-phase handlers not invoked unless auth is
          configured for the directory.

*   Common auth configuration stored in the core per-dir
          configuration; it has accessors `ap_auth_type`,
          `ap_auth_name`, and `ap_requires`.

*   Common routines, to handle the protocol end of things, at
          least for HTTP basic authentication
          (`ap_get_basic_auth_pw`, which sets the
          `connection-&gt;user` structure field
          automatically, and `ap_note_basic_auth_failure`,
          which arranges for the proper `WWW-Authenticate:`
          header to be sent back).

### &lt;a name="log_handlers" id="calibre_link-435" class="pcalibre1 calibre22 pcalibre3
    handlers&lt;/a&gt;

When a request has internally redirected, there is the question of
      what to log. Apache handles this by bundling the entire chain of redirects
      into a list of `request_rec` structures which are threaded
      through the `r-&gt;prev` and `r-&gt;next` pointers.
      The `request_rec` which is passed to the logging handlers in
      such cases is the one which was originally built for the initial request
      from the client; note that the `bytes_sent` field will only be
      correct in the last request in the chain (the one for which a response was
      actually sent).
```

`

Resource allocation and resource pools

One of the problems of writing and designing a server-pool server is that of preventing leakage, that is, allocating resources (memory, open files, *etc.*), without subsequently releasing them. The resource pool machinery is designed to make it easy to prevent this from happening, by allowing resource to be allocated in such a way that they are *automatically* released when the server is done with them.

The way this works is as follows: the memory which is allocated, file opened, *etc.*, to deal with a particular request are tied to a *resource pool* which is allocated for the request. The pool is a data structure which itself tracks the resources in question.

When the request has been processed, the pool is *cleared*. At that point, all the memory associated with it is released for reuse, all files associated with it are closed, and any other clean-up functions which are associated with the pool are run. When this is over, we can be confident that all the resource tied to the pool have been released, and that none of them have leaked.

Server restarts, and allocation of memory and resources for per-server configuration, are handled in a similar way. There is a *configuration pool*, which keeps track of resources which were allocated while reading the server configuration files, and handling the commands therein (for instance, the memory that was allocated for per-server module configuration, log files and other files that were opened, and so forth). When the server restarts, and has to reread the configuration files, the configuration pool is cleared, and so the memory and file descriptors which were taken up by reading them the last time are made available for reuse.

It should be noted that use of the pool machinery isn't generally obligatory, except for situations like logging handlers, where you really need to register cleanups to make sure that the log file gets closed when the server restarts (this is most easily done by using the function `ap_pfopen`, which also arranges for the underlying file descriptor to be closed before any child processes, such as for CGI scripts, are `exec`ed), or in case you are using the timeout machinery (which isn't yet even documented here). However, there are two benefits to using it: resources allocated to a pool never leak (even if you allocate a scratch string, and just forget about it); also, for memory allocation, `ap_palloc` is generally faster than `malloc`.

We begin here by describing how memory is allocated to pools, and then discuss how other resources are tracked by the resource pool machinery.

Allocation of memory in pools

Memory is allocated to pools by calling the function `ap_palloc`, which takes two arguments, one being a pointer to a resource pool structure, and the other being the amount of memory to allocate (in `char`s). Within handlers for handling requests, the most common way of getting a resource pool structure is by looking at the `pool` slot of the relevant `request_rec`; hence the repeated appearance of the following idiom in module code:

```
        int my_handler(request_rec *r)
        {
struct my_structure *foo;
          ...
          foo = (foo *)ap_palloc (r->pool, sizeof(my_structure));
        }
```

Note that *there is no* `ap_pfree` -- `ap_palloc`ed memory is freed only when the associated resource pool is cleared. This means that `ap_palloc` does not have to do as much accounting as `malloc()`; all it does in the typical case is to round up the size, bump a pointer, and do a range check.

(It also raises the possibility that heavy use of `ap_palloc` could cause a server process to grow excessively large. There are two ways to deal with this, which are dealt with below; briefly, you can use `malloc`, and try to be sure that all of the memory gets explicitly `free`d, or you can allocate a sub-pool of the main pool, allocate your memory in the sub-pool, and clear it out periodically. The latter technique is discussed in the section on sub-pools below, and is used in the directory-indexing code, in order to avoid excessive storage allocation when listing directories with thousands of files).

Allocating initialized memory

There are functions which allocate initialized memory, and are frequently useful. The function `ap_pcalloc` has the same interface as `ap_palloc`, but clears out the memory it allocates before it returns it. The function `ap_pstrdup` takes a resource pool and a `char *` as arguments, and allocates memory for a copy of the string the pointer points to, returning a pointer to the copy. Finally `ap_pstrcat` is a varargs-style function, which takes a pointer to a resource pool, and at least two `char *` arguments, the last of which must be `NULL`. It allocates enough memory to fit copies of each of the strings, as a unit; for instance:

```
        ap_pstrcat (r->pool, "foo", "/", "bar", NULL);
```

returns a pointer to 8 bytes worth of memory, initialized to `"foo/bar"`.

Commonly-used pools in the Apache Web server

A pool is really defined by its lifetime more than anything else. There are some static pools in http_main which are passed to various non-http_main functions as arguments at opportune times. Here they are:

`permanent_pool`

never passed to anything else, this is the ancestor of all pools

`pconf`

- subpool of permanent_pool
- created at the beginning of a config "cycle"; exists until the server is terminated or restarts; passed to all config-time routines, either via cmd->pool, or as the "pool *p" argument on those which don't take pools
- passed to the module init() functions

`ptemp`

- sorry I lie, this pool isn't called this currently in 1.3, I renamed it this in my pthreads development. I'm referring to the use of ptrans in the parent... contrast this with the later definition of ptrans in the child.
- subpool of permanent_pool
- created at the beginning of a config "cycle"; exists until the end of config parsing; passed to config-time routines *via* cmd->temp_pool. Somewhat of a "bastard child" because it isn't available everywhere. Used for temporary scratch space which may be needed by some config routines but which is deleted at the end of config.

`pchild`

- subpool of permanent_pool
- created when a child is spawned (or a thread is created); lives until that child (thread) is destroyed
- passed to the module child_init functions
- destruction happens right after the child_exit functions are called... (which may explain why I think child_exit is redundant and unneeded)

`ptrans`

- should be a subpool of pchild, but currently is a subpool of permanent_pool, see above
- cleared by the child before going into the accept() loop to receive a connection
- used as connection->pool

`r->pool`

- for the main request this is a subpool of connection->pool; for subrequests it is a subpool of the parent request's pool.
- exists until the end of the request (*i.e.*, ap_destroy_sub_req, or in child_main after process_request has finished)
- note that r itself is allocated from r->pool; *i.e.*, r->pool is first created and then r is the first thing palloc()d from it

For almost everything folks do, `r->pool` is the pool to use. But you can see how other lifetimes, such as pchild, are useful to some modules... such as modules that need to open a database connection once per child, and wish to clean it up when the child dies.

You can also see how some bugs have manifested themself, such as setting `connection->user` to a value from `r->pool` -- in this case connection exists for the lifetime of `ptrans`, which is longer than `r->pool` (especially if `r->pool` is a subrequest!). So the correct thing to do is to allocate from `connection->pool`.

And there was another interesting bug in `mod_include` / `mod_cgi`. You'll see in those that they do this test to decide if they should use `r->pool` or `r->main->pool`. In this case the resource that they are registering for cleanup is a child process. If it were registered in `r->pool`, then the code would `wait()` for the child when the subrequest finishes. With `mod_include` this could be any old `#include`, and the delay can be up to 3 seconds... and happened quite frequently. Instead the subprocess is registered in `r->main->pool` which causes it to be cleaned up when the entire request is done -- *i.e.*, after the output has been sent to the client and logging has happened.

Tracking open files, etc.

As indicated above, resource pools are also used to track other sorts of resources besides memory. The most common are open files. The routine which is typically used for this is `ap_pfopen`, which takes a resource pool and two strings as arguments; the strings are the same as the typical arguments to `fopen`, 例如，

```
...
FILE *f = ap_pfopen (r->pool, r->filename, "r");
if (f == NULL) { ... } else { ... }
```

There is also a `ap_popenf` routine, which parallels the lower-level `open` system call. Both of these routines arrange for the file to be closed when the resource pool in question is cleared.

Unlike the case for memory, there *are* functions to close files allocated with `ap_pfopen`, and `ap_popenf`, namely `ap_pfclose` and `ap_pclosef`. (This is because, on many systems, the number of files which a single process can have open is quite limited). It is important to use these functions to close files allocated with `ap_pfopen` and `ap_popenf`, since to do otherwise could cause fatal errors on systems such as Linux, which react badly if the same `FILE*` is closed more than once.

(Using the `close` functions is not mandatory, since the file will eventually be closed regardless, but you should consider it in cases where your module is opening, or could open, a lot of files).

Other sorts of resources -- cleanup functions

More text goes here. Describe the the cleanup primitives in terms of which the file stuff is implemented; also, `spawn_process`.

Pool cleanups live until `clear_pool()` is called: `clear_pool(a)` recursively calls `destroy_pool()` on all subpools of `a`; then calls all the cleanups for `a`; then releases all the memory for `a`. `destroy_pool(a)` calls `clear_pool(a)` and then releases the pool structure itself. *i.e.*, `clear_pool(a)` doesn't delete `a`, it just frees up all the resources and you can start using it again immediately.

Fine control -- creating and dealing with sub-pools, with a note on sub-requests

On rare occasions, too-free use of `ap_palloc()` and the associated primitives may result in undesirably profligate resource allocation. You can deal with such a case by creating a *sub-pool*, allocating within the sub-pool rather than the main pool, and clearing or destroying the sub-pool, which releases the resources which were associated with it. (This really *is* a rare situation; the only case in which it comes up in the standard module set is in case of listing directories, and then only with *very* large directories. Unnecessary use of the primitives discussed here can hair up your code quite a bit, with very little gain).

The primitive for creating a sub-pool is `ap_make_sub_pool`, which takes another pool (the parent pool) as an argument. When the main pool is cleared, the sub-pool will be destroyed. The sub-pool may also be cleared or destroyed at any time, by calling the functions `ap_clear_pool` and `ap_destroy_pool`, respectively. (The difference is that `ap_clear_pool` frees resources associated with the pool, while `ap_destroy_pool` also deallocates the pool itself. In the former case, you can allocate new resources within the pool, and clear it again, and so forth; in the latter case, it is simply gone).

One final note -- sub-requests have their own resource pools, which are sub-pools of the resource pool for the main request. The polite way to reclaim the resources associated with a sub request which you have allocated (using the `ap_sub_req_...` functions) is `ap_destroy_sub_req`, which frees the resource pool. Before calling this function, be sure to copy anything that you care about which might be allocated in the sub-request's resource pool into someplace a little less volatile (for instance, the filename in its `request_rec` structure).

(Again, under most circumstances, you shouldn't feel obliged to call this function; only 2K of memory or so are allocated for a typical sub request, and it will be freed anyway when the main request pool is cleared. It is only when you are allocating many, many sub-requests for a single main request that you should seriously consider the `ap_destroy_...` functions).

Configuration, commands and the like

One of the design goals for this server was to maintain external compatibility with the NCSA 1.3 server --- that is, to read the same configuration files, to process all the directives therein correctly, and in general to be a drop-in replacement for NCSA. On the other hand, another design goal was to move as much of the server's functionality into modules which have as little as possible to do with the monolithic server core. The only way to reconcile these goals is to move the handling of most commands from the central server into the modules.

However, just giving the modules command tables is not enough to divorce them completely from the server core. The server has to remember the commands in order to act on them later. That involves maintaining data which is private to the modules, and which can be either per-server, or per-directory. Most things are per-directory, including in particular access control and authorization information, but also information on how to determine file types from suffixes, which can be modified by `AddType` and `DefaultType` directives, and so forth. In general, the governing philosophy is that anything which *can* be made configurable by directory should be; per-server information is generally used in the standard set of modules for information like `Alias` es and `Redirect` s which come into play before the request is tied to a particular place in the underlying file system.

Another requirement for emulating the NCSA server is being able to handle the per-directory configuration files, generally called `.htaccess` files, though even in the NCSA server they can contain directives which have nothing at all to do with access control. Accordingly, after URI -> filename translation, but before performing any other phase, the server walks down the directory hierarchy of the underlying filesystem, following the translated pathname, to read any `.htaccess` files which might be present. The information which is read in then has to be *merged* with the applicable information from the server's own config files (either from the `<Directory>` sections in `access.conf` , or from defaults in `srm.conf` , which actually behaves for most purposes almost exactly like `<Directory />`).

Finally, after having served a request which involved reading `.htaccess` files, we need to discard the storage allocated for handling them. That is solved the same way it is solved wherever else similar problems come up, by tying those structures to the per-transaction resource pool.

Per-directory configuration structures

Let's look out how all of this plays out in `mod_mime.c` , which defines the file typing handler which emulates the NCSA server's behavior of determining file types from suffixes. What we'll be looking at, here, is the code which implements the `AddType` and `AddEncoding`

commands. These commands can appear in `.htaccess` files, so they must be handled in the module's private per-directory data, which in fact, consists of two separate tables for MIME types and encoding information, and is declared as follows:

```
typedef struct {
    table *forced_types;      /* Additional AddTyped stuff */
    table *encoding_types;    /* Added with AddEncoding... */
} mime_dir_config;
```

When the server is reading a configuration file, or `<Directory>` section, which includes one of the MIME module's commands, it needs to create a `mime_dir_config` structure, so those commands have something to act on. It does this by invoking the function it finds in the module's 'create per-dir config slot', with two arguments: the name of the directory to which this configuration information applies (or `NULL` for `srm.conf`), and a pointer to a resource pool in which the allocation should happen.

(If we are reading a `.htaccess` file, that resource pool is the per-request resource pool for the request; otherwise it is a resource pool which is used for configuration data, and cleared on restarts. Either way, it is important for the structure being created to vanish when the pool is cleared, by registering a cleanup on the pool if necessary).

For the MIME module, the per-dir config creation function just `ap_palloc`s the structure above, and a creates a couple of tables to fill it. That looks like this:

```
        void *create_mime_dir_config (pool *p, char *dummy)

        {

 mime_dir_config *new =

 (mime_dir_config *) ap_palloc (p, sizeof(mime_dir_config));

          new->forced_types = ap_make_table (p, 4);

          new->encoding_types = ap_make_table (p, 4);

          return new;
        }
```

Now, suppose we've just read in a `.htaccess` file. We already have the per-directory configuration structure for the next directory up in the hierarchy. If the `.htaccess` file we just read in didn't have any `AddType` or `AddEncoding` commands, its per-directory config structure for the MIME module is still valid, and we can just use it. Otherwise, we need to merge the two structures somehow.

To do that, the server invokes the module's per-directory config merge function, if one is present. That function takes three arguments: the two structures being merged, and a resource pool in which to allocate the result. For the MIME module, all that needs to be done is overlay the tables from the new per-directory config structure with those from the parent:

```
        void *merge_mime_dir_configs (pool *p, void *parent_dirv, void *subdirv)
        {
mime_dir_config *parent_dir = (mime_dir_config *)parent_dirv;
          mime_dir_config *subdir = (mime_dir_config *)subdirv;
          mime_dir_config *new =
(mime_dir_config *)ap_palloc (p, sizeof(mime_dir_config));
          new->forced_types = ap_overlay_tables (p, subdir->forced_types,
parent_dir->forced_types);
          new->encoding_types = ap_overlay_tables (p, subdir->encoding_types,
parent_dir->encoding_types);
          return new;
        }
```

As a note -- if there is no per-directory merge function present, the server will just use the subdirectory's configuration info, and ignore the parent's. For some modules, that works just fine (例如，for the includes module, whose per-directory configuration information consists solely of the state of the `XBITHACK`), and for those modules, you can just not declare one, and leave the corresponding structure slot in the module itself `NULL`.

Command handling

Now that we have these structures, we need to be able to figure out how to fill them. That involves processing the actual `AddType` and `AddEncoding` commands. To find commands, the server looks in the module's command table. That table contains information on how many arguments the commands take, and in what formats, where it is permitted, and so forth. That information is sufficient to allow the server to invoke most command-handling functions with pre-parsed arguments. Without further ado, let's look at the `AddType` command handler, which looks like this (the `AddEncoding` command looks basically the same, and won't be shown here):

```
        char *add_type(cmd_parms *cmd, mime_dir_config *m, char *ct, char *ext)
        {
if (*ext == '.') ++ext;
          ap_table_set (m->forced_types, ext, ct);
          return NULL;
        }
```

This command handler is unusually simple. As you can see, it takes four arguments, two of which are pre-parsed arguments, the third being the per-directory configuration structure for the module in question, and the fourth being a pointer to a `cmd_parms` structure. That

structure contains a bunch of arguments which are frequently of use to some, but not all, commands, including a resource pool (from which memory can be allocated, and to which cleanups should be tied), and the (virtual) server being configured, from which the module's per-server configuration data can be obtained if required.

Another way in which this particular command handler is unusually simple is that there are no error conditions which it can encounter. If there were, it could return an error message instead of `NULL`; this causes an error to be printed out on the server's `stderr`, followed by a quick exit, if it is in the main config files; for a `.htaccess` file, the syntax error is logged in the server error log (along with an indication of where it came from), and the request is bounced with a server error response (HTTP error status, code 500).

The MIME module's command table has entries for these commands, which look like this:

```
        command_rec mime_cmds[] = {

 { "AddType", add_type, NULL, OR_FILEINFO, TAKE2,

"a mime type followed by a file extension" },

          { "AddEncoding", add_encoding, NULL, OR_FILEINFO, TAKE2,

 "an encoding (例如, gzip), followed by a file extension" },
          { NULL }
        };
```

The entries in these tables are:

- The name of the command
- The function which handles it
- a `(void *)` pointer, which is passed in the `cmd_parms` structure to the command handler --- this is useful in case many similar commands are handled by the same function.
- A bit mask indicating where the command may appear. There are mask bits corresponding to each `AllowOverride` option, and an additional mask bit, `RSRC_CONF`, indicating that the command may appear in the server's own config files, but *not* in any `.htaccess` file.
- A flag indicating how many arguments the command handler wants pre-parsed, and how they should be passed in. `TAKE2` indicates two pre-parsed arguments. Other options are `TAKE1`, which indicates one pre-parsed argument, `FLAG`, which indicates that the argument should be `On` or `Off`, and is passed in as a boolean flag, `RAW_ARGS`, which causes the server to give the command the raw, unparsed arguments (everything but the command name itself). There is also `ITERATE`, which means that the handler looks the same as `TAKE1`, but that if multiple arguments are present, it should be called multiple times, and finally `ITERATE2`, which indicates that the command handler looks like a `TAKE2`, but if more arguments are present, then it should

be called multiple times, holding the first argument constant.
- Finally, we have a string which describes the arguments that should be present. If the arguments in the actual config file are not as required, this string will be used to help give a more specific error message. (You can safely leave this `NULL`).

Finally, having set this all up, we have to use it. This is ultimately done in the module's handlers, specifically for its file-typing handler, which looks more or less like this; note that the per-directory configuration structure is extracted from the `request_rec` 's per-directory configuration vector by using the `ap_get_module_config` function.

```
        int find_ct(request_rec *r)

        {

int i;

          char *fn = ap_pstrdup (r->pool, r->filename);

          mime_dir_config *conf = (mime_dir_config *)

ap_get_module_config(r->per_dir_config, &mime_module);
          char *type;

          if (S_ISDIR(r->finfo.st_mode)) {

r->content_type = DIR_MAGIC_TYPE;

            return OK;
          }

          if((i=ap_rind(fn,'.')) < 0) return DECLINED;

          ++i;

          if ((type = ap_table_get (conf->encoding_types, &fn[i])))

          {

r->content_encoding = type;

            /* go back to previous extension to try to use it as a type */

            fn[i-1] = '\0';

            if((i=ap_rind(fn,'.')) < 0) return OK;

            ++i;
          }

          if ((type = ap_table_get (conf->forced_types, &fn[i])))

          {

r->content_type = type;
          }

          return OK;
        }
```

Side notes -- per-server configuration, virtual servers, *etc*.

The basic ideas behind per-server module configuration are basically the same as those for per-directory configuration; there is a creation function and a merge function, the latter being invoked where a virtual server has partially overridden the base server configuration, and a combined structure must be computed. (As with per-directory configuration, the default if no merge function is specified, and a module is configured in some virtual server, is that the base configuration is simply ignored).

The only substantial difference is that when a command needs to configure the per-server private module data, it needs to go to the `cmd_parms` data to get at it. Here's an example, from the alias module, which also indicates how a syntax error can be returned (note that the per-directory configuration argument to the command handler is declared as a dummy, since the module doesn't actually have per-directory config data):

```
        char *add_redirect(cmd_parms *cmd, void *dummy, char *f, char *url)

        {

server_rec *s = cmd->server;

          alias_server_conf *conf = (alias_server_conf *)

ap_get_module_config(s->module_config,&alias_module);
          alias_entry *new = ap_push_array (conf->redirects);

          if (!ap_is_url (url)) return "Redirect to non-URL";

          new->fake = f; new->real = url;

          return NULL;
        }
```

````
````

Debugging Memory Allocation in APR

The allocation mechanisms within APR have a number of debugging modes that can be used to assist in finding memory problems. This document describes the modes available and gives instructions on activating them.

Available debugging options

Allocation Debugging - ALLOC_DEBUG

Debugging support: Define this to enable code which helps detect re-use of `free()` d memory and other such nonsense.

The theory is simple. The `FILL_BYTE` (`0xa5`) is written over all `malloc` 'd memory as we receive it, and is written over everything that we free up during a `clear_pool` . We check that blocks on the free list always have the `FILL_BYTE` in them, and we check during `palloc()` that the bytes still have `FILL_BYTE` in them. If you ever see garbage URLs or whatnot containing lots of `0xa5` s then you know something used data that's been freed or uninitialized.

Malloc Support - ALLOC_USE_MALLOC

If defined all allocations will be done with `malloc()` and `free()` d appropriately at the end.

This is intended to be used with something like Electric Fence or Purify to help detect memory problems. Note that if you're using efence then you should also add in `ALLOC_DEBUG` . But don't add in `ALLOC_DEBUG` if you're using Purify because `ALLOC_DEBUG` would hide all the uninitialized read errors that Purify can diagnose.

Pool Debugging - POOL_DEBUG

This is intended to detect cases where the wrong pool is used when assigning data to an object in another pool.

In particular, it causes the `table_{set,add,merge}n` routines to check that their arguments are safe for the `apr_table_t` they're being placed in. It currently only works with the unix multiprocess model, but could be extended to others.

Table Debugging - MAKE_TABLE_PROFILE

Provide diagnostic information about make_table() calls which are possibly too small.

This requires a recent gcc which supports `__builtin_return_address()`. The error_log output will be a message such as:

```
table_push: apr_table_t created by 0x804d874 hit limit of 10
```

Use `l *0x804d874` to find the source that corresponds to. It indicates that a `apr_table_t` allocated by a call at that address has possibly too small an initial `apr_table_t` size guess.

Allocation Statistics - ALLOC_STATS

Provide some statistics on the cost of allocations.

This requires a bit of an understanding of how `alloc.c` works.

Allowable Combinations

Not all the options outlined above can be activated at the same time. the following table gives more information.

ALLOC DEBUG	ALLOC USE MALLOC	POOL DEBUG	MAKE TABLE PROFILE	ALLOC STATS	
ALLOC DEBUG	-	No	Yes	Yes	Yes
ALLOC USE MALLOC	No	-	No	No	No
POOL DEBUG	Yes	No	-	Yes	Yes
MAKE TABLE PROFILE	Yes	No	Yes	-	Yes
ALLOC STATS	Yes	No	Yes	Yes	-

Additionally the debugging options are not suitable for multi-threaded versions of the server. When trying to debug with these options the server should be started in single process mode.

Activating Debugging Options

The various options for debugging memory are now enabled in the `apr_general.h` header file in APR. The various options are enabled by uncommenting the define for the option you wish to use. The section of the code currently looks like this (*contained in*

srclib/apr/include/apr_pools.h)

```
      /*
      #define ALLOC_DEBUG
      #define POOL_DEBUG
      #define ALLOC_USE_MALLOC
      #define MAKE_TABLE_PROFILE
      #define ALLOC_STATS
      */
      typedef struct ap_pool_t {
union block_hdr *first;
        union block_hdr *last;
        struct cleanup *cleanups;
        struct process_chain *subprocesses;
        struct ap_pool_t *sub_pools;
        struct ap_pool_t *sub_next;
        struct ap_pool_t *sub_prev;
        struct ap_pool_t *parent;
        char *free_first_avail;
      #ifdef ALLOC_USE_MALLOC
void *allocation_list;
      #endif
      #ifdef POOL_DEBUG
struct ap_pool_t *joined;
      #endif
int (*apr_abort)(int retcode);
        struct datastruct *prog_data;
      } ap_pool_t;
```

To enable allocation debugging simply move the `#define ALLOC_DEBUG` above the start of the comments block and rebuild the server.

Note

In order to use the various options the server **must** be rebuilt after editing the header file.

Documenting Apache 2.0

Apache 2.0 uses Doxygen to document the APIs and global variables in the the code. This will explain the basics of how to document using Doxygen.

Brief Description

To start a documentation block, use `/**` To end a documentation block, use `*/`

In the middle of the block, there are multiple tags we can use:

```
      Description of this functions purpose

      @param parameter_name description

      @return description

      @deffunc signature of the function
```

`deffunc` is not always necessary. DoxyGen does not have a full parser in it, so any prototype that use a macro in the return type declaration is too complex for scandoc. Those functions require a `deffunc`. An example (using > rather than >):

```
      /**
  * return the final element of the pathname
  * @param pathname The path to get the final element of
  * @return the final element of the path
  * @tip Examples:
  * <pre>
  *                 "/foo/bar/gum"   -> "gum"
  *                 "/foo/bar/gum/"  -> ""
  *                 "gum"            -> "gum"
  *                 "wi\\n32\\stuff" -> "stuff"
  * </pre>
  * @deffunc const char * ap_filename_of_pathname(const char *pathname)
  */
```

At the top of the header file, always include:

```
    /**
 * @package Name of library header
 */
```

Doxygen uses a new HTML file for each package. The HTML files are named {Name_of_library_header}.html, so try to be concise with your names.

For a further discussion of the possibilities please refer to the Doxygen site.

Apache 2.0 Hook Functions

Warning

This document is still in development and may be partially out of date.

In general, a hook function is one that Apache will call at some point during the processing of a request. Modules can provide functions that are called, and specify when they get called in comparison to other modules.

Creating a hook function

In order to create a new hook, four things need to be done:

Declare the hook function

Use the `AP_DECLARE_HOOK` macro, which needs to be given the return type of the hook function, the name of the hook, and the arguments. For example, if the hook returns an `int` and takes a `request_rec *` and an `int` and is called `do_something`, then declare it like this:

```
        AP_DECLARE_HOOK(int, do_something, (request_rec *r, int n))
```

This should go in a header which modules will include if they want to use the hook.

Create the hook structure

Each source file that exports a hook has a private structure which is used to record the module functions that use the hook. This is declared as follows:

```
        APR_HOOK_STRUCT(

APR_HOOK_LINK(do_something)

          ...
        )
```

Implement the hook caller

The source file that exports the hook has to implement a function that will call the hook. There are currently three possible ways to do this. In all cases, the calling function is called `ap_run_hookname()` .

Void hooks

If the return value of a hook is `void` , then all the hooks are called, and the caller is implemented like this:

```
AP_IMPLEMENT_HOOK_VOID(do_something, (request_rec *r, int n), (r, n))
```

The second and third arguments are the dummy argument declaration and the dummy arguments as they will be used when calling the hook. In other words, this macro expands to something like this:

```
void ap_run_do_something(request_rec *r, int n)
{
...
  do_something(r, n);
}
```

Hooks that return a value

If the hook returns a value, then it can either be run until the first hook that does something interesting, like so:

```
AP_IMPLEMENT_HOOK_RUN_FIRST(int, do_something, (request_rec *r, int n), (r, n),
```

The first hook that does *not* return `DECLINED` stops the loop and its return value is returned from the hook caller. Note that `DECLINED` is the tradition Apache hook return meaning "I didn't do anything", but it can be whatever suits you.

Alternatively, all hooks can be run until an error occurs. This boils down to permitting *two* return values, one of which means "I did something, and it was OK" and the other meaning "I did nothing". The first function that returns a value other than one of those two stops the loop, and its return is the return value. Declare these like so:

```
AP_IMPLEMENT_HOOK_RUN_ALL(int, do_something, (request_rec *r, int n), (r, n), O
```

Again, `OK` 和 `DECLINED` are the traditional values. You can use what you want.

Call the hook callers

At appropriate moments in the code, call the hook caller, like so:

```
        int n, ret;
        request_rec *r;
        ret=ap_run_do_something(r, n);
```

Hooking the hook

A module that wants a hook to be called needs to do two things.

Implement the hook function

Include the appropriate header, and define a static function of the correct type:

```
        static int my_something_doer(request_rec *r, int n)
        {
...
          return OK;
        }
```

Add a hook registering function

During initialisation, Apache will call each modules hook registering function, which is included in the module structure:

```
        static void my_register_hooks()
        {
ap_hook_do_something(my_something_doer, NULL, NULL, APR_HOOK_MIDDLE);
        }
        mode MODULE_VAR_EXPORT my_module =
        {
...
          my_register_hooks       /* register hooks */
        };
```

Controlling hook calling order

In the example above, we didn't use the three arguments in the hook registration function that control calling order. There are two mechanisms for doing this. The first, rather crude, method, allows us to specify roughly where the hook is run relative to other modules. The final argument control this. There are three possible values: `APR_HOOK_FIRST` , `APR_HOOK_MIDDLE` 和 `APR_HOOK_LAST` .

All modules using any particular value may be run in any order relative to each other, but, of course, all modules using `APR_HOOK_FIRST` will be run before `APR_HOOK_MIDDLE` which are before `APR_HOOK_LAST` . Modules that don't care when they are run should use `APR_HOOK_MIDDLE` . *(I spaced these out so people could do stuff like `APR_HOOK_FIRST-2` to get in slightly earlier, but is this wise? - Ben)*

Note that there are two more values, `APR_HOOK_REALLY_FIRST` 和 `APR_HOOK_REALLY_LAST` . These should only be used by the hook exporter.

The other method allows finer control. When a module knows that it must be run before (or after) some other modules, it can specify them by name. The second (third) argument is a NULL-terminated array of strings consisting of the names of modules that must be run before (after) the current module. For example, suppose we want "mod_xyz.c" and "mod_abc.c" to run before we do, then we'd hook as follows:

```
        static void register_hooks()

        {

  static const char * const aszPre[] = { "mod_xyz.c", "mod_abc.c", NULL };

          ap_hook_do_something(my_something_doer, aszPre, NULL, APR_HOOK_MIDDLE);
        }
```

Note that the sort used to achieve this is stable, so ordering set by `APR_HOOK_ORDER` is preserved, as far as is possible.

<cite class="calibre27">Ben Laurie</cite>, 15th August 1999

Converting Modules from Apache 1.3 to Apache 2.0

This is a first attempt at writing the lessons I learned when trying to convert the `mod_mmap_static` module to Apache 2.0. It's by no means definitive and probably won't even be correct in some ways, but it's a start.

The easier changes ...

Cleanup Routines

These now need to be of type `apr_status_t` and return a value of that type. Normally the return value will be `APR_SUCCESS` unless there is some need to signal an error in the cleanup. Be aware that even though you signal an error not all code yet checks and acts upon the error.

Initialisation Routines

These should now be renamed to better signify where they sit in the overall process. So the name gets a small change from `mmap_init` to `mmap_post_config` . The arguments passed have undergone a radical change and now look like

- `apr_pool_t *p`
- `apr_pool_t *plog`
- `apr_pool_t *ptemp`
- `server_rec *s`

Data Types

A lot of the data types have been moved into the APR. This means that some have had a name change, such as the one shown above. The following is a brief list of some of the changes that you are likely to have to make.

- `pool` becomes `apr_pool_t`
- `table` becomes `apr_table_t`

The messier changes...

Register Hooks

The new architecture uses a series of hooks to provide for calling your functions. These you'll need to add to your module by way of a new function, `static void register_hooks(void)`. The function is really reasonably straightforward once you understand what needs to be done. Each function that needs calling at some stage in the processing of a request needs to be registered, handlers do not. There are a number of phases where functions can be added, and for each you can specify with a high degree of control the relative order that the function will be called in.

This is the code that was added to `mod_mmap_static`:

```
static void register_hooks(void)
{
    static const char * const aszPre[]={ "http_core.c",NULL };
    ap_hook_post_config(mmap_post_config,NULL,NULL,HOOK_MIDDLE);
    ap_hook_translate_name(mmap_static_xlat,aszPre,NULL,HOOK_LAST);
};
```

This registers 2 functions that need to be called, one in the `post_config` stage (virtually every module will need this one) and one for the `translate_name` phase. note that while there are different function names the format of each is identical. So what is the format?

```
        ap_hook_<var class="calibre40">phase_name</var>(<var class="calibre40">function_n
        <var class="calibre40">predecessors</var>, <var class="calibre40">successors</var
```

There are 3 hook positions defined...

- `HOOK_FIRST`
- `HOOK_MIDDLE`
- `HOOK_LAST`

To define the position you use the position and then modify it with the predecessors and successors. Each of the modifiers can be a list of functions that should be called, either before the function is run (predecessors) or after the function has run (successors).

In the `mod_mmap_static` case I didn't care about the `post_config` stage, but the `mmap_static_xlat` **must** be called after the core module had done it's name translation, hence the use of the aszPre to define a modifier to the position `HOOK_LAST`.

Module Definition

There are now a lot fewer stages to worry about when creating your module definition. The old defintion looked like

```
module MODULE_VAR_EXPORT <var class="calibre27">module_name</var>_module =
{
    STANDARD_MODULE_STUFF,
    /* initializer */
    /* dir config creater */
    /* dir merger --- default is to override */
    /* server config */
    /* merge server config */
    /* command handlers */
    /* handlers */
    /* filename translation */
    /* check_user_id */
    /* check auth */
    /* check access */
    /* type_checker */
    /* fixups */
    /* logger */
    /* header parser */
    /* child_init */
    /* child_exit */
    /* post read-request */
};
```

The new structure is a great deal simpler...

```
module MODULE_VAR_EXPORT <var class="calibre27">module_name</var>_module =
{
    STANDARD20_MODULE_STUFF,
    /* create per-directory config structures */
    /* merge per-directory config structures  */
    /* create per-server config structures    */
    /* merge per-server config structures     */
    /* command handlers */
    /* handlers */
    /* register hooks */
};
```

Some of these read directly across, some don't. I'll try to summarise what should be done below.

The stages that read directly across :

`/* dir config creater */`

`/* create per-directory config structures */`

`/* server config */`

`/* create per-server config structures */`

`/* dir merger */`

`/* merge per-directory config structures */`

`/* merge server config */`

`/* merge per-server config structures */`

`/* command table */`

`/* command apr_table_t */`

`/* handlers */`

`/* handlers */`

The remainder of the old functions should be registered as hooks. There are the following hook stages defined so far...

`ap_hook_post_config`

this is where the old `_init` routines get registered

`ap_hook_http_method`

retrieve the http method from a request. (legacy)

`ap_hook_open_logs`

open any specified logs

`ap_hook_auth_checker`

check if the resource requires authorization

`ap_hook_access_checker`

check for module-specific restrictions

`ap_hook_check_user_id`

check the user-id and password

`ap_hook_default_port`

retrieve the default port for the server

`ap_hook_pre_connection`

do any setup required just before processing, but after accepting

`ap_hook_process_connection`

run the correct protocol

`ap_hook_child_init`

call as soon as the child is started

`ap_hook_create_request`

??

`ap_hook_fixups`

last chance to modify things before generating content

`ap_hook_handler`

generate the content

`ap_hook_header_parser`

lets modules look at the headers, not used by most modules, because they use `post_read_request` for this

`ap_hook_insert_filter`

to insert filters into the filter chain

`ap_hook_log_transaction`

log information about the request

`ap_hook_optional_fn_retrieve`

retrieve any functions registered as optional

`ap_hook_post_read_request`

called after reading the request, before any other phase

`ap_hook_quick_handler`

called before any request processing, used by cache modules.

`ap_hook_translate_name`

translate the URI into a filename

`ap_hook_type_checker`

determine and/or set the doc type

Request Processing in Apache 2.0

Warning

Warning - this is a first (fast) draft that needs further revision!

Several changes in Apache 2.0 affect the internal request processing mechanics. Module authors need to be aware of these changes so they may take advantage of the optimizations and security enhancements.

The first major change is to the subrequest and redirect mechanisms. There were a number of different code paths in Apache 1.3 to attempt to optimize subrequest or redirect behavior. As patches were introduced to 2.0, these optimizations (and the server behavior) were quickly broken due to this duplication of code. All duplicate code has been folded back into `ap_process_request_internal()` to prevent the code from falling out of sync again.

This means that much of the existing code was 'unoptimized'. It is the Apache HTTP Project's first goal to create a robust and correct implementation of the HTTP server RFC. Additional goals include security, scalability and optimization. New methods were sought to optimize the server (beyond the performance of Apache 1.3) without introducing fragile or insecure code.

The Request Processing Cycle

All requests pass through `ap_process_request_internal()` in `request.c`, including subrequests and redirects. If a module doesn't pass generated requests through this code, the author is cautioned that the module may be broken by future changes to request processing.

To streamline requests, the module author can take advantage of the hooks offered to drop out of the request cycle early, or to bypass core Apache hooks which are irrelevant (and costly in terms of CPU.)

The Request Parsing Phase

Unescapes the URL

The request's `parsed_uri` path is unescaped, once and only once, at the beginning of internal request processing.

This step is bypassed if the proxyreq flag is set, or the `parsed_uri.path` element is unset. The module has no further control of this one-time unescape operation, either failing to unescape or multiply unescaping the URL leads to security reprecussions.

Strips Parent and This Elements from the URI

All `/../` 和 `/./` elements are removed by `ap_getparents()`. This helps to ensure the path is (nearly) absolute before the request processing continues.

This step cannot be bypassed.

Initial URI Location Walk

Every request is subject to an `ap_location_walk()` call. This ensures that `<Location>` sections are consistently enforced for all requests. If the request is an internal redirect or a sub-request, it may borrow some or all of the processing from the previous or parent request's ap_location_walk, so this step is generally very efficient after processing the main request.

translate_name

Modules can determine the file name, or alter the given URI in this step. For example, `mod_vhost_alias` will translate the URI's path into the configured virtual host, `mod_alias` will translate the path to an alias path, and if the request falls back on the core, the `DocumentRoot` is prepended to the request resource.

If all modules `DECLINE` this phase, an error 500 is returned to the browser, and a "couldn't translate name" error is logged automatically.

Hook: map_to_storage

After the file or correct URI was determined, the appropriate per-dir configurations are merged together. For example, `mod_proxy` compares and merges the appropriate `<Proxy>` sections. If the URI is nothing more than a local (non-proxy) `TRACE` request, the core handles the request and returns `DONE`. If no module answers this hook with `OK` 或 `DONE`, the core will run the request filename against the `<Directory>` 和 `<Files>` sections. If the request 'filename' isn't an absolute, legal filename, a note is set for later termination.

URI Location Walk

Every request is hardened by a second `ap_location_walk()` call. This reassures that a translated request is still subjected to the configured `<Location>` sections. The request again borrows some or all of the processing from its previous `location_walk` above, so this step is almost always very efficient unless the translated URI mapped to a substantially different path or Virtual Host.

Hook: header_parser

The main request then parses the client's headers. This prepares the remaining request processing steps to better serve the client's request.

The Security Phase

Needs Documentation. Code is:

```
switch (ap_satisfies(r)) {
case SATISFY_ALL:
case SATISFY_NOSPEC:
    if ((access_status = ap_run_access_checker(r)) != 0) {
        return decl_die(access_status, "check access", r);
    }

    if (ap_some_auth_required(r)) {
        if (((access_status = ap_run_check_user_id(r)) != 0)
            || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type(r)
                          ? "check user.  No user file?"
                          : "perform authentication. AuthType not set!",
                          r);
        }

        if (((access_status = ap_run_auth_checker(r)) != 0)
            || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type(r)
                          ? "check access.  No groups file?"
                          : "perform authentication. AuthType not set!",
                          r);
        }
    }
    break;

case SATISFY_ANY:
    if (((access_status = ap_run_access_checker(r)) != 0)) {
        if (!ap_some_auth_required(r)) {
            return decl_die(access_status, "check access", r);
        }

        if (((access_status = ap_run_check_user_id(r)) != 0)
            || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type(r)
                          ? "check user.  No user file?"
                          : "perform authentication. AuthType not set!",
                          r);
        }

        if (((access_status = ap_run_auth_checker(r)) != 0)
            || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type(r)
                          ? "check access.  No groups file?"
                          : "perform authentication. AuthType not set!",
                          r);
        }
    }
    break;
}
```

The Preparation Phase

Hook: type_checker

The modules have an opportunity to test the URI or filename against the target resource, and set mime information for the request. Both `mod_mime` 和 `mod_mime_magic` use this phase to compare the file name or contents against the administrator's configuration and set the content type, language, character set and request handler. Some modules may set up their filters or other request handling parameters at this time.

If all modules `DECLINE` this phase, an error 500 is returned to the browser, and a "couldn't find types" error is logged automatically.

Hook: fixups

Many modules are 'trounced' by some phase above. The fixups phase is used by modules to 'reassert' their ownership or force the request's fields to their appropriate values. It isn't always the cleanest mechanism, but occasionally it's the only option.

The Handler Phase

This phase is **not** part of the processing in `ap_process_request_internal()`. Many modules prepare one or more subrequests prior to creating any content at all. After the core, or a module calls `ap_process_request_internal()` it then calls `ap_invoke_handler()` to generate the request.

Hook: insert_filter

Modules that transform the content in some way can insert their values and override existing filters, such that if the user configured a more advanced filter out-of-order, then the module can move its order as need be. There is no result code, so actions in this hook better be trusted to always succeed.

Hook: handler

The module finally has a chance to serve the request in its handler hook. Note that not every prepared request is sent to the handler hook. Many modules, such as `mod_autoindex`, will create subrequests for a given URI, and then never serve the subrequest, but simply lists it for the user. Remember not to put required teardown from the hooks above into this module, but register pool cleanups against the request pool to free resources as required.

How filters work in Apache 2.0

Warning

This is a cut 'n paste job from an email (<022501c1c529$f63a9550$7f00000a@KOJ>) and only reformatted for better readability. It's not up to date but may be a good start for further research.

Filter Types

There are three basic filter types (each of these is actually broken down into two categories, but that comes later).

`CONNECTION`

Filters of this type are valid for the lifetime of this connection. (`AP_FTYPE_CONNECTION` , `AP_FTYPE_NETWORK`)

`PROTOCOL`

Filters of this type are valid for the lifetime of this request from the point of view of the client, this means that the request is valid from the time that the request is sent until the time that the response is received. (`AP_FTYPE_PROTOCOL` , `AP_FTYPE_TRANSCODE`)

`RESOURCE`

Filters of this type are valid for the time that this content is used to satisfy a request. For simple requests, this is identical to `PROTOCOL` , but internal redirects and sub-requests can change the content without ending the request. (`AP_FTYPE_RESOURCE` , `AP_FTYPE_CONTENT_SET`)

It is important to make the distinction between a protocol and a resource filter. A resource filter is tied to a specific resource, it may also be tied to header information, but the main binding is to a resource. If you are writing a filter and you want to know if it is resource or protocol, the correct question to ask is: "Can this filter be removed if the request is redirected to a different resource?" If the answer is yes, then it is a resource filter. If it is no, then it is most likely a protocol or connection filter. I won't go into connection filters, because they seem to be well understood. With this definition, a few examples might help:

Byterange

We have coded it to be inserted for all requests, and it is removed if not used. Because this filter is active at the beginning of all requests, it can not be removed if it is redirected, so this is a protocol filter.

http_header

This filter actually writes the headers to the network. This is obviously a required filter (except in the asis case which is special and will be dealt with below) and so it is a protocol filter.

Deflate

The administrator configures this filter based on which file has been requested. If we do an internal redirect from an autoindex page to an index.html page, the deflate filter may be added or removed based on config, so this is a resource filter.

The further breakdown of each category into two more filter types is strictly for ordering. We could remove it, and only allow for one filter type, but the order would tend to be wrong, and we would need to hack things to make it work. Currently, the `RESOURCE` filters only have one filter type, but that should change.

How are filters inserted?

This is actually rather simple in theory, but the code is complex. First of all, it is important that everybody realize that there are three filter lists for each request, but they are all concatenated together. So, the first list is `r->output_filters` , then `r->proto_output_filters` , and finally `r->connection->output_filters` . These correspond to the `RESOURCE` , `PROTOCOL` , and `CONNECTION` filters respectively. The problem previously, was that we used a singly linked list to create the filter stack, and we started from the "correct" location. This means that if I had a `RESOURCE` filter on the stack, and I added a `CONNECTION` filter, the `CONNECTION` filter would be ignored. This should make sense, because we would insert the connection filter at the top of the `c->output_filters` list, but the end of `r->output_filters` pointed to the filter that used to be at the front of `c->output_filters` . This is obviously wrong. The new insertion code uses a doubly linked list. This has the advantage that we never lose a filter that has been inserted. Unfortunately, it comes with a separate set of headaches.

The problem is that we have two different cases were we use subrequests. The first is to insert more data into a response. The second is to replace the existing response with an internal redirect. These are two different cases and need to be treated as such.

In the first case, we are creating the subrequest from within a handler or filter. This means that the next filter should be passed to `make_sub_request` function, and the last resource filter in the sub-request will point to the next filter in the main request. This makes sense, because the sub-request's data needs to flow through the same set of filters as the main request. A graphical representation might help:

```
Default_handler --> includes_filter --> byterange --> ...
```

If the includes filter creates a sub request, then we don't want the data from that sub-request to go through the includes filter, because it might not be SSI data. So, the subrequest adds the following:

```
Default_handler --> includes_filter -/-> byterange --> ...
                                    /
Default_handler --> sub_request_core
```

What happens if the subrequest is SSI data? Well, that's easy, the `includes_filter` is a resource filter, so it will be added to the sub request in between the `Default_handler` and the `sub_request_core` filter.

The second case for sub-requests is when one sub-request is going to become the real request. This happens whenever a sub-request is created outside of a handler or filter, and NULL is passed as the next filter to the `make_sub_request` function.

In this case, the resource filters no longer make sense for the new request, because the resource has changed. So, instead of starting from scratch, we simply point the front of the resource filters for the sub-request to the front of the protocol filters for the old request. This means that we won't lose any of the protocol filters, neither will we try to send this data through a filter that shouldn't see it.

The problem is that we are using a doubly-linked list for our filter stacks now. But, you should notice that it is possible for two lists to intersect in this model. So, you do you handle the previous pointer? This is a very difficult question to answer, because there is no "right" answer, either method is equally valid. I looked at why we use the previous pointer. The only reason for it is to allow for easier addition of new servers. With that being said, the solution I chose was to make the previous pointer always stay on the original request.

This causes some more complex logic, but it works for all cases. My concern in having it move to the sub-request, is that for the more common case (where a sub-request is used to add data to a response), the main filter chain would be wrong. That didn't seem like a good idea to me.

Asis

The final topic. :-) Mod_Asis is a bit of a hack, but the handler needs to remove all filters except for connection filters, and send the data. If you are using `mod_asis`, all other bets are off.

Explanations

The absolutely last point is that the reason this code was so hard to get right, was because we had hacked so much to force it to work. I wrote most of the hacks originally, so I am very much to blame. However, now that the code is right, I have started to remove some hacks. Most people should have seen that the `reset_filters` 和 `add_required_filters` functions are gone. Those inserted protocol level filters for error conditions, in fact, both functions did the same thing, one after the other, it was really strange. Because we don't lose protocol filters for error cases any more, those hacks went away. The `HTTP_HEADER`, `Content-length`, and `Byterange` filters are all added in the `insert_filters` phase, because if they were added earlier, we had some interesting interactions. Now, those could all be moved to be inserted with the `HTTP_IN`, `CORE`, and `CORE_IN` filters. That would make the code easier to follow.

Apache 2.0 Thread Safety Issues

When using any of the threaded mpms in Apache 2.0 it is important that every function called from Apache be thread safe. When linking in 3rd party extensions it can be difficult to determine whether the resulting server will be thread safe. Casual testing generally won't tell you this either as thread safety problems can lead to subtle race conditons that may only show up in certain conditions under heavy load.

Global and static variables

When writing your module or when trying to determine if a module or 3rd party library is thread safe there are some common things to keep in mind.

First, you need to recognize that in a threaded model each individual thread has its own program counter, stack and registers. Local variables live on the stack, so those are fine. You need to watch out for any static or global variables. This doesn't mean that you are absolutely not allowed to use static or global variables. There are times when you actually want something to affect all threads, but generally you need to avoid using them if you want your code to be thread safe.

In the case where you have a global variable that needs to be global and accessed by all threads, be very careful when you update it. If, for example, it is an incrementing counter, you need to atomically increment it to avoid race conditions with other threads. You do this using a mutex (mutual exclusion). Lock the mutex, read the current value, increment it and write it back and then unlock the mutex. Any other thread that wants to modify the value has to first check the mutex and block until it is cleared.

If you are using APR, have a look at the `apr_atomic_*` functions and the `apr_thread_mutex_*` functions.

errno

This is a common global variable that holds the error number of the last error that occurred. If one thread calls a low-level function that sets errno and then another thread checks it, we are bleeding error numbers from one thread into another. To solve this, make sure your module or library defines `_REENTRANT` or is compiled with `-D_REENTRANT`. This will make errno a per-thread variable and should hopefully be transparent to the code. It does this by doing something like this:

```
#define errno (*(__errno_location()))
```

which means that accessing errno will call `__errno_location()` which is provided by the libc. Setting `_REENTRANT` also forces redefinition of some other functions to their `*_r` equivalents and sometimes changes the common `getc` / `putc` macros into safer function calls. Check your libc documentation for specifics. Instead of, or in addition to `_REENTRANT` the symbols that may affect this are `_POSIX_C_SOURCE` , `_THREAD_SAFE` , `_SVID_SOURCE` , and `_BSD_SOURCE` .

Common standard troublesome functions

Not only do things have to be thread safe, but they also have to be reentrant. `strtok()` is an obvious one. You call it the first time with your delimiter which it then remembers and on each subsequent call it returns the next token. Obviously if multiple threads are calling it you will have a problem. Most systems have a reentrant version of of the function called `strtok_r()` where you pass in an extra argument which contains an allocated `char *` which the function will use instead of its own static storage for maintaining the tokenizing state. If you are using APR you can use `apr_strtok()` .

`crypt()` is another function that tends to not be reentrant, so if you run across calls to that function in a library, watch out. On some systems it is reentrant though, so it is not always a problem. If your system has `crypt_r()` chances are you should be using that, or if possible simply avoid the whole mess by using md5 instead.

Common 3rd Party Libraries

The following is a list of common libraries that are used by 3rd party Apache modules. You can check to see if your module is using a potentially unsafe library by using tools such as `ldd(1)` 和 `nm(1)` . For PHP, for example, try this:

```
% ldd libphp4.so
libsablot.so.0 => /usr/local/lib/libsablot.so.0 (0x401f6000)
libexpat.so.0 => /usr/lib/libexpat.so.0 (0x402da000)
libsnmp.so.0 => /usr/lib/libsnmp.so.0 (0x402f9000)
libpdf.so.1 => /usr/local/lib/libpdf.so.1 (0x40353000)
libz.so.1 => /usr/lib/libz.so.1 (0x403e2000)
libpng.so.2 => /usr/lib/libpng.so.2 (0x403f0000)
libmysqlclient.so.11 => /usr/lib/libmysqlclient.so.11 (0x40411000)
libming.so => /usr/lib/libming.so (0x40449000)
libm.so.6 => /lib/libm.so.6 (0x40487000)
libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x404a8000)
libjpeg.so.62 => /usr/lib/libjpeg.so.62 (0x404e7000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x40505000)
libssl.so.2 => /lib/libssl.so.2 (0x40532000)
libcrypto.so.2 => /lib/libcrypto.so.2 (0x40560000)
libresolv.so.2 => /lib/libresolv.so.2 (0x40624000)
libdl.so.2 => /lib/libdl.so.2 (0x40634000)
libnsl.so.1 => /lib/libnsl.so.1 (0x40637000)
libc.so.6 => /lib/libc.so.6 (0x4064b000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
```

In addition to these libraries you will need to have a look at any libraries linked statically into the module. You can use `nm(1)` to look for individual symbols in the module.

Library List

Please drop a note to dev@httpd.apache.org if you have additions or corrections to this list.

Library	Version	Thread Safe?	
ASpell/PSpell	?		
Berkeley DB	3.x, 4.x	Yes	Be c thre
bzip2	Yes	Both low-level and high-level APIs are thread-safe. However, high-level API requires thread-safe access to errno.	
cdb	?		
		c-client uses `strtok()` 和 `gethostbyname()` which	

C-Client	Perhaps	are not thread-safe on most C library implementations. c-client's static data is meant to be shared across threads. If `strtok()` 和 `gethostbyname()` are thread-safe on your OS, c-client *may* be thread-safe.	
cpdflib	?		
libcrypt	?		
Expat	Yes	Need a separate parser instance per thread	
FreeTDS	?		
FreeType	?		
GD 1.8.x	?		
GD 2.0.x	?		
gdbm	No	Errors returned via a static `gdbm_error` variable	
ImageMagick	5.2.2	Yes	Ima vers
Imlib2	?		
libjpeg	v6b	?	
libmysqlclient	Yes	Use mysqlclient_r library variant to ensure thread-safety. For more information, please read http://www.mysql.com/doc/en/Threaded_clients.html.	
Ming	0.2a	?	
Net-SNMP	5.0.x	?	
OpenLDAP	2.1.x	Yes	Use safe
OpenSSL	0.9.6g	Yes	Req `CRY` `CRY`
liboci8 (Oracle 8+)	8.x,9.x	?	
pdflib	5.0.x	Yes	PDF indi V1. http
libpng	1.0.x	?	
libpng	1.2.x	?	
libpq (PostgreSQL)	7.x	Yes	Don watc
Sablotron	0.95	?	

zlib	1.1.4	Yes	Reli func whic

词汇和索引

词汇表

此词汇表包含了与Apache相关的一些常用术语的详细定义，以及对网络服务的一般说明，并提供了相关的更详细资料的连接。

定义

访问控制(Access Control)

对网络领域访问的限制。对Apache来说，通常是指对某些*URL*访问的限制。参见：认证、授权、访问控制

算法(Algorithm)

通过有限步骤解决问题的一个明确的公式或者一套规则。用于加密的算法通常称为<dfn class="calibre27">加密算法(Cipher)</dfn>。

Apache扩展工具(APache eXtension Tool) (apxs)

一个perl脚本，用于编译模块(module)源代码为动态共享对象(DSO)，并帮助安装到Apache web服务器中。 参见：`apxs`

Apache可移植运行时(Apache Portable Runtime) (APR)

一组在操作系统和服务器之间提供许多基本接口的库。APR是一个与Apache HTTP Server平行的独立项目。 参见：Apache Portable Runtime Project

认证(Authentication)

对诸如服务器、客户端或用户等网络实体的真实性鉴定。 参见：认证、授权、访问控制

证书(Certificate)

用于鉴别诸如服务器或客户端的网络实体的一个数据记录。一个证书包含有：若干其所有者的X.509信息片段(称为主题[subject])，其签发的证书机构(Certification Authority)(称为发行者[issuer])，还有其所有者的公钥(public key)和由证书机构(CA)制作的签名。网络实体将用CA证书校验这些签名。 参见：SSL/TLS 加密

证书签发请求(Certificate Signing Request) (CSR)

一个提交给证书机构(Certification Authority)的用其CA证书的私钥(Private Key)签名的未经签发的证书(certificate)。一旦这个CSR被签发，则成为一个真正的证书。 参见：SSL/TLS 加密

证书机构(Certification Authority) (CA)

一个旨在对已经通过保密方法得到鉴定的网络实体签发证书的可信的第三方团体。其他网络实体可以通过验证签名来确定一个证书的持有人是否通过了CA的鉴定。 参见：SSL/TLS 加密

加密算法(Cipher)

一种用于数据加密的算法或系统。如：DES、IDEA、RC4等等。 参见：SSL/TLS 加密

密文(Ciphertext)

明文(Plaintext)通过加密算法(Cipher)处理后的结果。 参见：SSL/TLS 加密

公共网关接口(Common Gateway Interface) (CGI)

一种允许在web服务器和外部程序之间使用外部程序响应请求的接口的标准定义。此接口最早由(美国)国家计算机安全协会(NCSA)制定，另外还有一个RFC项目。 参见：CGI动态页面

配置指令(Configuration Directive)

参见：指令(Directive)

配置文件(Configuration File)

一个控制Apache配置的含有若干指令(Directives)的文本文件。 参见：配置文件

连接(CONNECT)

一种通过HTTP通道代理原始数据的HTTP方法(method)。可以用于封装其他协议，如SSL协议。

作用域(Context)

指令(Directives)在配置文件中的许可区域。 参见：描述指令的术语

数字签名(Digital Signature)

验证证书或其他文件的一个经过加密的文本块。证书机构(Certification Authority)对公钥 *(Public Key)*生成一个散列并嵌入证书 *(Certificate)*，以建立签名，然后用其自身的私钥 *(Private Key)*加密这个散列。只有证书机构(CA)的公钥才能解密此签名，以证实持有此证书的网络实体已经通过了CA的鉴定。 参见：SSL/TLS 加密

指令(Directive)

位于配置文件(Configuration File)中的控制一个或多个Apache行为的配置命令。 参见：指令索引

动态共享对象(Dynamic Shared Object) (DSO)

与Apache `httpd` 二进制映象分开编译的可以在被调用时加载的模块(Modules)。 参见：动态共享对象支持

环境变量(Environment Variable) (env-variable)

由操作系统shell管理的用于存储信息和程序之间通讯的已命名的变量。Apache存储的内部变量有时也称作环境变量，但它们是被存储在Apache内部结构中的，而不是存储在shell环境中的。 参见：Apache的环境变量

出口限制(Export-Crippled)

降低加密强度(和安全度)以符合美国出口监管条例(EAR)的规定。出口限制加密的软件被限制只能使用密钥长度较短的密钥，从而使密文可以被暴力破解。 参见：SSL/TLS 加密

过滤器(Filter)

服务器用来接收和发送数据的过程。输入过滤器处理客户端发送到服务器的数据，而输出过滤器处理服务器发送到客户端的文档。比如，`INCLUDES` 输出过滤器处理服务器端包含(Server Side Includes)文档。 参见：过滤器

全称域名(Fully-Qualified Domain-Name) (FQDN)

网络实体的唯一的名称，由主机名和域名组成，并能够被解析为一个IP地址。比如，`www` 是一个主机名，`example.com` 是一个域名，那么 `www.example.com` 就是一个全称域名。

处理器(Handler)

处理器是一个文件被调用时，Apache所执行操作的内部表现。一般来说，文件都有基于其文件类型的隐含处理器。通常，文件都只是被服务器作简单的提交，只有某些文件类型被特殊地"处理"。比如：`cgi-script` 处理器使文件作为CGI脚本被处理。 参见：Apache处理器的使用

散列/哈希(Hash)

一个从变长字符串生成定长字符串的单向算法。不同的输入字符串一般会产生不同的输出散列值(取决于hash函数)。

头(Header)

在实际内容之前发送的HTTP请求和响应的一部分，其中包含描述内容的元信息(meta-information)。

.htaccess

网站目录树结构中的一个配置文件(configuration file)，使配置指令(Directive)作用于其所在目录及其所有子目录。此文件可以包含几乎所有类型的指令，而并不仅仅是其文件名所暗示的访问控制指令。 参见：配置文件

httpd.conf

Apache的主配置文件(configuration file)，默认值是 `/usr/local/apache2/conf/httpd.conf` ，但可以通过运行时或编译时的配置改变。 参见：配置文件

超文本传输协议(HyperText Transfer Protocol) (HTTP)

在WWW上使用的标准传输协议。Apache实现了此协议的1.1版本，即在RFC 2616中定义的HTTP/1.1 。

HTTPS

安全(Secure)超文本传输协议，在WWW上使用的标准加密通讯机制。实质上，它是SSL基础上的HTTP。 参见：SSL/TLS 加密

方法(Method)

对于HTTP，是由客户端在请求行中指定的对一个资源执行的操作。HTTP中的方法有诸如 `GET` 、 `POST` 、 `PUT` 等等。

消息摘要(Message Digest)

消息的一个散列值，可以用于校验消息内容是否在传输过程中有所改变。 参见：SSL/TLS 加密

MIME类型(MIME-type)

描述被传输文档的类型的一种方法。因其格式借用了多用途网际邮件扩展(MIME)而得名。由以斜杠分隔的一个主类型和一个副类型组成。例如： `text/html` ， `image/gif` ， `application/octet-stream` 。在HTTP中，MIME类型包含在 `Content-Type` 头(header)中被传输。 参见：mod_mime

模块(Module)

程序的一个独立的部分。Apache中的多数功能都包含在模块中以供取舍。被编译进入Apache `httpd` 二进制映象的模块称为<dfn class="calibre27">静态模块(static module)</dfn>，而单独存储的可以有选择地在运行时被加载的模块称为<dfn class="calibre27">动态模块(dynamic module)</dfn>或DSO。默认被包含的模块称为<dfn class="calibre27">基本模块(base module)</dfn>。很多Apache可以使用的模块都不是作为Apache HTTP服务器tar包(tarball)的一部分发行的，这些模块被称为<dfn class="calibre27">第三方模块(third-party module)</dfn>。 参见：模块索引

模块幻数(Module Magic Number) (MMN)

模块幻数是Apache源代码中定义的与模块二进制映象兼容性相关的常量。当Apache内部结构、函数调用和API的重要部分发生改变，再也不能保证此二进制映象的兼容性的时候，这个值会被改变。一旦MMN被改变，所有的第三方模块必须至少被重新编译，有时候甚至要修改源代码才能在Apache的新版本中运行。

OpenSSL

开源的SSL/TLS工具包 参见：http://www.openssl.org/

通行码(Pass Phrase)

保护私钥文件的一个词或短语，以避免被未授权用户加密。通常，它只是密码算法(Cipher)中保密的用于加密/解密的密钥。 参见：SSL/TLS 加密

明文(Plaintext)

未加密的文本。

私钥(Private Key)

公钥加密系统中保密的密钥，用于对到来的消息解密和对外出的消息签名。 参见：SSL/TLS 加密

代理(Proxy)

处于客户端和原始服务器*(origin server)*之间的中间服务器。它接收来自客户端的请求，传输到原始服务器，并把原始服务器的响应返回给客户端。如果几个客户端请求的内容相同，代理可以从其缓存中取出此内容，而不必每次都从原始服务器读取，从而缩短了响应时间。 参见：mod_proxy

公钥(Public Key)

公钥加密系统中对公众公开的密钥，用于加密送往其持有者的消息和解密由其持有者制作的签名。 参见：SSL/TLS 加密

公钥加密系统(Public Key Cryptography)

对使用一个加密密钥和一个解密密钥的不对称加密系统的研究和应用。相应的一对这样的密钥组成了密钥对。也称为"非对称加密系统"(Asymmetric Cryptography) 参见：SSL/TLS 加密

正则表达式(Regular Expression) (Regex)

一种对模式的文字表述，比如，"所有以字母A开头的单词"，"每个10位的电话号码" 还可以是"每个包含两个逗号，而且没有大写字母Q的句子"。正则表达式在Apache中非常有用，可以非常灵活地对一组文件或资源应用某种属性，例如，任何"images"目录下的.gif 和.jpg 文件可以表述为" `/images/.*(jpg|gif)$` "。Apache使用的是由PCRE库提供的Perl兼容的正则表达式。

反向代理(Reverse Proxy)

一个在客户端看来是原始服务器*(origin server)*的代理(proxy)服务器。出于安全考虑或为了实现均衡负载，借此对客户端隐藏原始服务器。

安全套接字层(Secure Sockets Layer) (SSL)

由Netscape公司建立的，在TCP/IP网络中实现常规通讯认证和加密的协议。它被广泛地用于*HTTPS*，即SSL基础上的超文本传输协议。 参见：SSL/TLS 加密

服务器端包含(Server Side Includes) (SSI)

在HTML文件中嵌入处理指令的一种技术。 参见：服务器端包含简介

会话(Session)

一般是指一个通讯的上下文信息。

SSLeay

由Eric A. Young开发的最初的SSL/TLS的实现库

对称密码系统(Symmetric Cryptography)

使用单个密钥执行加密和解密操作的密码算法研究和应用。 参见：SSL/TLS 加密

Tar包(Tarball)

用 `tar` 工具收集的文件包。Apache发行版是存储在用tar或pkzip压缩的文件中的。

传输层安全(Transport Layer Security) (TLS)

Internet工程任务组(IETF)建立的SSL的后续协议，在TCP/IP网络中实现常规通讯认证和加密。TLS的版本1和SSL的版本3基本一致。 参见：SSL/TLS 加密

统一资源定位器(Uniform Resource Locator) (URL)

资源在Internet中的名称/地址。它是正式名称为统一资源标识符(Uniform Resource Identifier)的非正式称呼。URL通常由一个类型，比如 `http` 或 `https` ，一个主机名和一个路径组成。指向本页面的URL是 `http://httpd.apache.org/docs/2.2/glossary.html`

统一资源标识符(Uniform Resource Identifier) (URI)

识别一个抽象或者物理资源的简洁字符串。由RFC 2396正式定义。互联网上使用的URI通常也被称为URL

虚拟主机(Virtual Hosting)

使用单个Apache实例提供多个网站。基于*IP*的虚拟主机*(IP virtual hosting)*基于IP区分各网站，而基于名称的虚拟主机*(name-based virtual hosting)*按主机名区分，从而在同一个IP地址上宿主多个网站。 参见：Apache虚拟主机文档

X.509

由国际电信联盟(ITU)推荐的用于SSL/TLS认证的一种认证证书类型。 参见：SSL/TLS 加密

指令索引

这里列示了Apache标准发行版中的所有指令。指令的描述采用统一的格式，其中用到的缩略语在指令术语字典有详细说明。

指令速查以概要形式提供了每个指令的细节。

- AcceptFilter
- AcceptMutex
- AcceptPathInfo
- AccessFileName
- Action
- AddAlt
- AddAltByEncoding
- AddAltByType
- AddCharset
- AddDefaultCharset
- AddDescription
- AddEncoding
- AddHandler
- AddIcon
- AddIconByEncoding
- AddIconByType
- AddInputFilter
- AddLanguage
- AddModuleInfo
- AddOutputFilter
- AddOutputFilterByType
- AddType
- Alias
- AliasMatch
- Allow
- AllowCONNECT
- AllowEncodedSlashes
- AllowOverride
- Anonymous
- Anonymous_LogEmail
- Anonymous_MustGiveEmail
- Anonymous_NoUserID

指令速查

指令速查列出了每个指令的用法、默认值、状态和作用域。更详细的说明可以查阅指令字典。

第一列是指令的名称和用法； 第二列是指令的默认值(如果存在)，如果很长写不下，则后面跟有"+"表示截断。 第三和第四列是指令允许出现的位置和状态，其中缩略语如下表所示：

s	server config
v	virtual host
d	directory
h	.htaccess
C	核心模块
M	MPM
B	基本模块
E	扩展模块
X	试验模块

AcceptFilter protocol accept_filter	s	C	
根据协议类型对监听Socket进行优化			
[AcceptMutex Default	method](#calibre_link-610)	Default	s
Apache用于串行化多个子进程在(多个)网络套接字(socket)上接受请求的方法			
[AcceptPathInfo On	Off	Default](#calibre_link-238)	De
是否接受附带多余路径名信息的请求			
AccessFileName filename [filename] ...	.htaccess	sv	C
分布式配置文件的名字			
Action action-type cgi-script [virtual]	svdh	B	
针对特定的处理器或内容类型激活一个CGI脚本			

AddAlt string file [file] ...	svdh	B	
Alternate text to display for a file, instead of an icon selected by filename			
AddAltByEncoding string MIME-encoding [MIME-encoding] ...	svdh	B	
Alternate text to display for a file instead of an icon selected by MIME-encoding			
AddAltByType string MIME-type [MIME-type] ...	svdh	B	
Alternate text to display for a file, instead of an icon selected by MIME content-type			
AddCharset charset extension [extension] ...	svdh	B	
在给定的文件扩展名与特定的字符集之间建立映射			
[AddDefaultCharset On	Off	charset] (#calibre_link-328)	Off
当应答内容是text/plain或text/html时，在HTTP应答头中加入的默认字符集			
AddDescription string file [file] ...	svdh	B	
Description to display for a file			
AddEncoding MIME-enc extension [extension] ...	svdh	B	
在文件扩展名与特定的编码方式之间建立映射关系			
AddHandler handler-name extension [extension] ...	svdh	B	
在文件扩展名与特定的处理器之间建立映射			
AddIcon icon name [name] ...	svdh	B	
Icon to display for a file selected by name			
AddIconByEncoding icon MIME-encoding [MIME-encoding] ...	svdh	B	
Icon to display next to files selected			

by MIME content-encoding			
AddIconByType icon MIME-type [MIME-type] ...	svdh	B	
Icon to display next to files selected by MIME content-type			
AddInputFilter filter[;filter...] extension [extension] ...	svdh	B	
在文件扩展名与特定的输入过滤器之间建立映射			
AddLanguage MIME-lang extension [extension] ...	svdh	B	
在文件扩展名与特定的语言之间建立映射			
AddModuleInfo module-name string	sv	E	
为server-info处理器显示的模块增加额外信息			
AddOutputFilter filter[;filter...] extension [extension] ...	svdh	B	
在文件扩展名与特定的输出过滤器之间建立映射关系			
AddOutputFilterByType filter[;filter...] MIME-type [MIME-type] ...	svdh	C	
对特定的MIME类型指定输出过滤器			
AddType MIME-type extension [extension] ...	svdh	B	
在给定的文件扩展名与特定的内容类型之间建立映射			
[Alias URL-path file-path	directory-path] (#calibre_link-220)	sv	B
映射URL到文件系统的特定区域			
[AliasMatch regex file-path	directory-path] (#calibre_link-512)	sv	B
使用正则表达式映射URL到文件系统			
[Allow from all	host	env=env-variable [host	en ...] 58
控制哪些主机能够访问服务器的该区域			
AllowCONNECT port [port] ...	443 563	sv	E

通过代理允许CONNECT的端口号			
[AllowEncodedSlashes On	Off](#calibre_link-733)	Off	sv
确定是否允许URL中使用经过编码的路径分割符			
[AllowOverride All	None	directive-type directive-type] ...	All
确定允许存在于.htaccess文件中的指令类型			
Anonymous user [user] ...	dh	E	
Specifies userIDs that are allowed access without password verification			
[Anonymous_LogEmail On	Off](#calibre_link-735)	On	dh
Sets whether the password entered will be logged in the error log			
[Anonymous_MustGiveEmail On	Off](#calibre_link-736)	On	dh
Specifies whether blank passwords are allowed			
[Anonymous_NoUserID On	Off](#calibre_link-737)	Off	dh
Sets whether the userID field may be empty			
[Anonymous_VerifyEmail On	Off](#calibre_link-738)	Off	dh
Sets whether to check the password field for a correctly formatted email address			
[AuthBasicAuthoritative On	Off](#calibre_link-739)	On	dh
指定是否将(基本)认证和授权操作交由更底层的模块来处理			
AuthBasicProvider provider-name [provider-name] ...	file	dh	B
设置该区域的(基本)认证支持者(Provider)			
AuthDBDUserPWQuery query	d	E	
SQL query to look up a password for a user			
AuthDBDUserRealmQuery query	d	E	

SQL query to look up a password hash for a user and realm.			
AuthDBMGroupFile file-path	dh	E	
Sets the name of the database file containing the list of user groups for authorization			
[AuthDBMType default	SDBM	GDBM	ND
Sets the type of database file that is used to store passwords			
AuthDBMUserFile file-path	dh	E	
Sets the name of a database file containing the list of users and passwords for authentication			
[AuthDefaultAuthoritative On	Off](#calibre_link-745)	On	dh
指定是否将认证操作交由更底层的模块来处理			
[AuthDigestAlgorithm MD5	MD5-sess](#calibre_link-746)	MD5	dh
选择在摘要认证中用于计算请求和应答的散列值的算法			
AuthDigestDomain URI [URI] ...	dh	X	
在同一保护区域中需要进行摘要认证的URI			
[AuthDigestNcCheck On	Off](#calibre_link-748)	Off	s
Enables or disables checking of the nonce-count sent by the server			
AuthDigestNonceFormat format	dh	X	
Determines how the nonce is generated			
AuthDigestNonceLifetime seconds	300	dh	X
服务器nonce(当前值)的有效秒数			
AuthDigestProvider provider-name [provider-name] ...	file	dh	X
设置该区域的(摘要)认证支持者(Provider)			
[AuthDigestQop none	auth	auth-int [auth	au (#c

			75
指定摘要认证的保护质量			
AuthDigestShmemSize size	1000	s	X
为了跟踪客户端而分配的共享内存字节数			
AuthGroupFile file-path	dh	B	
设定一个包含用于执行用户认证的用户组列表的纯文本文件			
AuthLDAPBindDN *distinguished-name*	dh	E	
Optional DN to use in binding to the LDAP server			
AuthLDAPBindPassword *password*	dh	E	
Password used in conjuction with the bind DN			
AuthLDAPCharsetConfig *file-path*	s	E	
Language to charset conversion configuration file			
[AuthLDAPCompareDNOnServer on	off](#calibre_link-757)	on	dh
Use the LDAP server to compare the DNs			
[AuthLDAPDereferenceAliases never	searching	finding	alw (#c 75
When will the module de-reference aliases			
AuthLDAPGroupAttribute *attribute*	dh	E	
LDAP attributes used to check for group membership			
[AuthLDAPGroupAttributeIsDN on	off](#calibre_link-760)	on	dh
Use the DN of the client username when checking for group membership			
[AuthLDAPRemoteUserIsDN on	off](#calibre_link-761)	off	dh
Use the DN of the client username to set the REMOTE_USER environment variable			

[AuthLDAPUrl _url [NONE	SSL	TLS	ST (#c 76
URL specifying the LDAP search parameters			
AuthName auth-domain	dh	C	
用于HTTP认证的授权域			
<AuthnProviderAlias baseProvider Alias> ... </AuthnProviderAlias>	sv	E	
封装一组定义扩展认证支持者的指令，并为其指定一个别名			
[AuthType Basic	Digest](#calibre_link-323)	dh	C
用户认证类型			
AuthUserFile file-path	dh	B	
设定一个含有认证使用的用户名/密码列表的纯文本文件			
[AuthzDBMAuthoritative On	Off](#calibre_link-764)	On	dh
Sets whether authorization will be passed on to lower level modules			
[AuthzDBMType default	SDBM	GDBM	ND
Sets the type of database file that is used to store list of user groups			
[AuthzDefaultAuthoritative On	Off](#calibre_link-766)	On	dh
指定是否将授权操作交由更底层的模块来处理			
[AuthzGroupFileAuthoritative On	Off](#calibre_link-767)	On	dh
指定是否将授权操作交由更底层的模块来处理			
[AuthzLDAPAuthoritative on	off](#calibre_link-768)	on	dh
Prevent other authentication modules from authenticating the user if this one fails			
[AuthzOwnerAuthoritative On	Off](#calibre_link-769)	On	dh
指定是否将授权操作交由更底层的模块来处理			

[AuthzUserAuthoritative On	Off](#calibre_link-770)	On	dh
指定是否将授权操作交由更底层的模块来处理			
BrowserMatch *regex [!]env-variable*[=*value*] [[!]*env-variable*[=*value*]] ...	svdh	B	
基于User-Agent头有条件地设置环境变量			
BrowserMatchNoCase *regex [!]env-variable*[=*value*] [[!]*env-variable*[=*value*]] ...	svdh	B	
基于不区分大小写的User-Agent头有条件地设置环境变量			
[BufferedLogs On	Off](#calibre_link-771)	Off	s
在将日志写入磁盘前先在内存中进行缓冲			
CacheDefaultExpire seconds	3600 (one hour)	sv	E
The default duration to cache a document when no expiry date is specified.			
CacheDirLength length	2	sv	E
The number of characters in subdirectory names			
CacheDirLevels levels	3	sv	E
The number of levels of subdirectories in the cache.			
CacheDisable url-string	sv	E	
Disable caching of specified URLs			
CacheEnable cache_type url-string	sv	E	
Enable caching of specified URLs using a specified storage manager			
CacheFile file-path [file-path] ...	s	X	
Cache a list of file handles at startup time			
[CacheIgnoreCacheControl On	Off](#calibre_link-772)	Off	sv
Ignore request to not serve cached content to client			
CacheIgnoreHeaders header-string	None	sv	E

[header-string] ...	None	sv	E
Do not store the given HTTP header(s) in the cache.			
[CacheIgnoreNoLastMod On	Off](#calibre_link-461)	Off	sv
Ignore the fact that a response has no Last Modified header.			
CacheLastModifiedFactor float	0.1	sv	E
The factor used to compute an expiry date based on the LastModified date.			
CacheMaxExpire seconds	86400 (one day)	sv	E
The maximum time in seconds to cache a document			
CacheMaxFileSize bytes	1000000	sv	E
The maximum size (in bytes) of a document to be placed in the cache			
CacheMinFileSize bytes	1	sv	E
The minimum size (in bytes) of a document to be placed in the cache			
[CacheNegotiatedDocs On	Off](#calibre_link-239)	Off	sv
允许经过内容协商的文档被代理服务器缓存			
CacheRoot directory	sv	E	
The directory root under which cache files are stored			
[CacheStoreNoStore On	Off](#calibre_link-463)	Off	sv
Attempt to cache requests or responses that have been marked as no-store.			
[CacheStorePrivate On	Off](#calibre_link-462)	Off	sv
Attempt to cache responses that the server has marked as private			
CGIMapExtension cgi-path .extension	dh	C	
定位CGI脚本解释器			
CharsetDefault charset	svdh	X	
Charset to translate into			

CharsetOptions option [option] ...	DebugLevel=0 NoImpl +	svdh	X
Configures charset translation behavior			
CharsetSourceEnc charset	svdh	X	
Source charset of files			
[CheckSpelling on	off](#calibre_link-513)	Off	svc
Enables the spelling module			
[ContentDigest On	Off](#calibre_link-779)	Off	svc
允许生成Content-MD5应答头			
CookieDomain *domain*	svdh	E	
The domain to which the tracking cookie applies			
CookieExpires *expiry-period*	svdh	E	
Expiry time for the tracking cookie			
CookieLog filename	sv	B	
设定针对cookies的日志文件名			
CookieName *token*	Apache	svdh	E
Name of the tracking cookie			
[CookieStyle _Netscape	Cookie	Cookie2	RF
Format of the cookie header field			
[CookieTracking on	off](#calibre_link-627)	off	svc
Enables tracking cookie			
CoreDumpDirectory directory	s	M	
Apache使用的内核转储目录			
[CustomLog file	pipe format	nickname env=[!]environment-variable]	sv
设定日志的文件名和格式			
[Dav On	Off	provider-name](#calibre_link-781)	Of
Enable WebDAV HTTP methods			

Enable WebDAV HTTP methods			
[DavDepthInfinity on	off](#calibre_link-782)	off	svd
Allow PROPFIND, Depth: Infinity requests			
DavGenericLockDB file-path	svd	E	
Location of the DAV lock database			
DavLockDB file-path	sv	E	
Location of the DAV lock database			
DavMinTimeout seconds	0	svd	E
Minimum amount of time the server holds a lock on a DAV resource			
DBDExptime time-in-seconds	sv	E	
Keepalive time for idle connections			
DBDKeep number	sv	E	
Maximum sustainednumber of connections			
DBDMax number	sv	E	
Maximum number of connections			
DBDMin number	sv	E	
Minimum number of connections			
DBDParams param1=value1[,param2=value2]	sv	E	
Parameters for database connection			
[DBDPersist 0	1](#calibre_link-790)	sv	E
Whether to use persistent connections			
DBDPrepareSQL "SQL statement" label	sv	E	
Define an SQL prepared statement			
DBDriver name	sv	E	
Specify an SQL driver			
DefaultIcon url-path	svdh	B	
Icon to display for files when no specific icon is configured			

为所有文件设定特定的默认语言			
DefaultType MIME-type	text/plain	svdh	C
在服务器无法由其他方法确定内容类型时，发送的默认MIME内容类型			
DeflateBufferSize value	8096	sv	E
用于zlib一次压缩的片断大小(字节)			
DeflateCompressionLevel value	sv	E	
将输出内容压缩的程度			
DeflateFilterNote [type] notename	sv	E	
在日志中放置压缩率标记			
DeflateMemLevel value	9	sv	E
zlib在压缩时最多可以使用多少内存			
DeflateWindowSize value	15	sv	E
Zlib压缩窗口(compression window)的大小			
[Deny from all	host	env=env-variable [host	en ...] 58
控制哪些主机被禁止访问服务器			
<Directory directory-path> ... </Directory>	sv	C	
封装一组指令，使之仅对文件空间中的某个目录及其子目录生效			
DirectoryIndex local-url [local-url] ...	index.html	svdh	B
当客户端请求一个目录时寻找的资源列表			
<DirectoryMatch regex> ... </DirectoryMatch>	sv	C	
封装一些指令并作用于文件系统中匹配正则表达式的所有目录及其子目录			
[DirectorySlash On	Off](#calibre_link-799)	On	sv
打开或关闭目录结尾斜线(/)自动补全功能			
DocumentRoot directory-path	/usr/local/apache/h +	sv	C
组成网络上可见的主文档树的根目录			

[DumpIOInput On	Off](#calibre_link-800)	Off	s
将所有输入的内容记录到错误日志中			
[DumpIOOutput On	Off](#calibre_link-801)	Off	s
将所有输出的内容记录到错误日志中			
[EnableExceptionHook On	Off](#calibre_link-802)	Off	s
在子进程崩溃以后启用一个钩子来运行异常处理程序			
[EnableMMAP On	Off](#calibre_link-605)	On	svc
在递送中使用内存映射(memory-mapping)来读取文件			
[EnableSendfile On	Off](#calibre_link-606)	On	svc
使用操作系统内核的sendfile支持来将文件发送到客户端			
ErrorDocument error-code document	svdh	C	
当遇到错误的时候服务器将给客户端什么样的应答			
[ErrorLog file-path	syslog:facility]	logs/error_log (Uni +	sv
存放错误日志的位置			
Example	svdh	X	
Demonstration directive to illustrate the Apache module API			
[ExpiresActive On	Off](#calibre_link-804)	svdh	E
启用或禁用产生"Expires:"和"Cache-Control:"头的功能			
ExpiresByType MIME-type <code>seconds	svdh	E	
由MIME类型配置的Expires头的值			
ExpiresDefault <code>seconds	svdh	E	
默认有效期的计算方法			
[ExtendedStatus On	Off](#calibre_link-807)	Off	s
Keep track of extended status information for each request			
ExtFilterDefine filtername parameters	s	E	

Define an external filter			
ExtFilterOptions option [option] ...	DebugLevel=0 NoLogS +	d	E
Configure `mod_ext_filter` options			
FileETag component ...	INode MTime Size	svdh	C
用以创建ETag应答头的文件的属性			
<Files filename> ... </Files>	svdh	C	
包含作用于匹配指定文件名的指令			
<FilesMatch regex> ... </FilesMatch>	svdh	C	
包含作用于与正则表达式匹配的文件名的指令			
FilterChain [+=-@!]filter-name ...	svdh	B	
Configure the filter chain			
FilterDeclare filter-name [type]	svdh	B	
Declare a smart filter			
FilterProtocol filter-name [provider-name] proto-flags	svdh	B	
Deal with correct HTTP protocol handling			
[FilterProvider filter-name provider-name [req	resp	env]=dispatch match] (#calibre_link-63)	svc
Register a content filter			
FilterTrace filter-name level	svd	B	
Get debug/diagnostic information from `mod_filter`			
[ForceLanguagePriority None	Prefer	Fallback [Prefer	Fa (#c 37
指定无法匹配单个文档的情况下所采取的动作			
[ForceType MIME-type	None](#calibre_link-342)	dh	C
强制所有匹配的文件被作为指定的MIME类型进行伺服			
[ForensicLog filename	pipe](#calibre_link-497)	sv	E

Sets filename of the forensic log			
GracefulShutDownTimeout seconds	s	M	
指定优雅停止服务器的超时秒数			
Group unix-group	#-1	s	M
对请求提供服务的Apache子进程运行时的用户组			
[Header [condition] set	append	add	un
配置HTTP应答头			
HeaderName filename	svdh	B	
Name of the file that will be inserted at the top of the index listing			
[HostnameLookups On	Off	Double](#calibre_link-493)	Of
启用对客户端IP的DNS查找			
[IdentityCheck On	Off](#calibre_link-810)	Off	sv
启用对远端用户的RFC1413身份鉴定的日志			
IdentityCheckTimeout seconds	30	svd	E
Determines the timeout duration for ident requests			
<IfDefine [!]parameter-name> ... </IfDefine>	svdh	C	
封装一组只有在启动时当测试结果为真时才生效的指令			
[<IfModule [!]module-file	module-identifier> ... </IfModule>](#calibre_link-447)	svdh	C
封装指令并根据指定的模块是否启用为条件而决定是否进行处理			
<IfVersion [[!]operator] version> ... </IfVersion>	svdh	E	
contains version dependent configuration			
[ImapBase map	referer	URL](#calibre_link-99)	htt

Default `base` for imagemap files			
[ImapDefault error	nocontent	map	ref
Default action when an imagemap is called with coordinates that are not explicitly mapped			
[ImapMenu none	formatted	semiformatted	un (#c 10
Action if no coordinates are given when calling an imagemap			
[Include file-path	directory-path] (#calibre_link-32)	svd	C
在服务器配置文件中包含其它配置文件			
IndexIgnore file [file] ...	svdh	B	
Adds to the list of files to hide when listing a directory			
[IndexOptions [+	-]option [[+	-]option] ...] (#calibre_link-241)	svc
Various configuration settings for directory indexing			
[IndexOrderDefault Ascending	Descending Name	Date	Siz
Sets the default ordering of the directory index			
IndexStyleSheet url-path	svdh	B	
Adds a CSS stylesheet to the directory index			
[ISAPIAppendLogToErrors on	off](#calibre_link-159)	off	svc
把ISAPI扩展的 HSE_APPEND_LOG_PARAMETER 请求记录在错误日志中			
[ISAPIAppendLogToQuery on	off](#calibre_link-158)	on	svc
把ISAPI扩展的 HSE_APPEND_LOG_PARAMETER 请求记录在查询域中			

ISAPICacheFile file-path [file-path] ...	sv	B	
启动时载入的ISAPI动态连接库			
[ISAPIFakeAsync on	off](#calibre_link-813)	off	svd
为ISAPI回调模拟异步支持			
[ISAPILogNotSupported on	off](#calibre_link-814)	off	svd
记录ISAPI不支持的功能调用			
ISAPIReadAheadBuffer size	49152	svdh	B
传送到ISAPI扩展的预读缓冲区大小			
[KeepAlive On	Off](#calibre_link-50)	On	sv
启用HTTP持久链接			
KeepAliveTimeout seconds	5	sv	C
持久链接中服务器在两次请求之间等待的秒数			
LanguagePriority MIME-lang [MIME-lang] ...	svdh	B	
在客户端没有指示语言偏好的情况下，语言变体的优先级列表			
LDAPCacheEntries number	1024	s	E
主LDAP缓冲的最大条目数			
LDAPCacheTTL seconds	600	s	E
search/bind缓冲项目有效时限			
LDAPConnectionTimeout seconds	s	E	
指定套接字连接超时秒数			
LDAPOpCacheEntries number	1024	s	E
LDAP compare缓冲区的大小			
LDAPOpCacheTTL seconds	600	s	E
操作缓冲有效时限			
LDAPSharedCacheFile directory-path/filename	s	E	
设置共享内存缓冲区文件			
LDAPSharedCacheSize bytes	102400	s	E
共享内存缓冲区的字节大小			

directory-path/filename/nickname [password]	svdh	E	
Sets the file containing or nickname referring to a per connection client certificate. Not all LDAP toolkits support per connection client certificates.			
LDAPTrustedGlobalCert type directory-path/filename [password]	s	E	
Sets the file or database containing global trusted Certificate Authority or global client certificates			
LDAPTrustedMode type	svdh	E	
Specifies the SSL/TLS mode to be used when connecting to an LDAP server.			
[LDAPVerifyServerCert On	Off](#calibre_link-818)	On	s
Force server certificate verification			
<Limit method [method] ... > ... </Limit>	svdh	C	
仅对指定的HTTP方法进行访问控制			
<LimitExcept method [method] ... > ... </LimitExcept>	svdh	C	
对除了指定方法以外的所有HTTP方法进行访问控制			
LimitInternalRecursion number [number]	10	sv	C
指定内部重定向和嵌套子请求的最大数量			
LimitRequestBody bytes	0	svdh	C
限制客户端发送的HTTP请求体的最大字节长度			
LimitRequestFields number	100	s	C
限制接受客户端请求中HTTP请求头域的数量			
LimitRequestFieldsize bytes	s	C	
限制客户端发送的请求头的字节数			
LimitRequestLine bytes	8190	s	C

限制接受客户端发送的HTTP请求行的字节数			
LimitXMLRequestBody bytes	1000000	svdh	C
限制基于XML的请求体的大小			
Listen [IP-address:]portnumber [protocol]	s	M	
服务器监听的IP地址和端口			
ListenBacklog backlog	s	M	
半链接(pending connection)队列的最大长度			
LoadFile *filename* [*filename*] ...	s	E	
加载已命名的目标文件或库			
LoadModule *module filename*	s	E	
加载目标文件或库，并将其添加到活动模块列表			
[<Location URL-path	URL> ... </Location>] (#calibre_link-118)	sv	C
将封装的指令作用于匹配的URL			
<LocationMatch regex> ... </LocationMatch>	sv	C	
将封装的指令作用于正则表达式匹配的URL			
LockFile filename	logs/accept.lock	s	M
接受串行锁文件的位置			
[LogFormat format	nickname nickname]	"%h %l %u %t \"%r\" +	sv
定义访问日志的记录格式			
LogLevel level	warn	sv	C
控制错误日志的详细程度			
MaxClients number	s	M	
允许同时伺服的最大接入请求数量			
MaxKeepAliveRequests number	100	sv	C
一个持久链接中允许的最大请求数量			
MaxMemFree KBytes	0	s	M

况下允许持有的最大自由内存数量(KB)			
MaxRequestsPerChild number	10000	s	M
每个子进程在其生存期内允许伺服的最大请求数量			
MaxRequestsPerThread number	0	s	M
Limit on the number of requests that an individual thread will handle during its life			
MaxSpareServers number	10	s	M
空闲子进程的最大数量			
MaxSpareThreads number	s	M	
最大空闲线程数			
MaxThreads number	2048	s	M
Set the maximum number of worker threads			
MCacheMaxObjectCount value	1009	s	E
最大缓存对象数			
MCacheMaxObjectSize bytes	10000	s	E
缓存允许的最大文档大小(字节)			
MCacheMaxStreamingBuffer size_in_bytes	the smaller of 1000 +	s	E
内存中允许缓冲的最大流式响应字节长度			
MCacheMinObjectSize bytes	0	s	E
允许缓存的最小文档大小(字节)			
[MCacheRemovalAlgorithm LRU	GDSF](#calibre_link-827)	GDSF	s
定义在需要时哪个文档被移出缓存的算法			
MCacheSize KBytes	100	s	E
缓存允许使用的最大内存量，以KB为单位			
MetaDir directory	.web	svdh	E
Name of the directory to find CERN-style meta information files			

[MetaFiles on	off](#calibre_link-830)	off	svc
Activates CERN meta-file processing			
MetaSuffix suffix	.meta	svdh	E
File name suffix for the file containg CERN-style meta information			
MimeMagicFile file-path	sv	E	
使用特定的Magic文件激活根据文件内容确定文件MIME类型的功能			
MinSpareServers number	5	s	M
空闲子进程的最小数量			
MinSpareThreads number	s	M	
最小空闲线程数			
MMapFile file-path [file-path] ...	s	X	
Map a list of files into memory at startup time			
[ModMimeUsePathInfo On	Off](#calibre_link-832)	Off	d
将path_info当成是文件名的一个组成部分			
[MultiviewsMatch Any	NegotiatedOnly	Filters	Ha [Ha
在使用MultiViews查询所匹配的文件时要包含的文件类型			
NameVirtualHost addr[:port]	s	C	
为一个基于域名的虚拟主机指定一个IP地址(和端口)			
NoProxy host [host] ...	sv	E	
直接进行连接的主机/域/网络			
NWSSLTrustedCerts filename [filename] ...	s	B	
附加的客户端证书列表			
NWSSLUpgradeable [IP-address:]portnumber	s	B	
允许在请求中将一个连接升级为SSL连接			

[Options [+	-]option [[+	(#calibre_link-153)	All
配置在特定目录中可以使用哪些特性			
Order ordering	Deny,Allow	dh	B
控制默认的访问状态与Allow和Deny指令生效的顺序			
PassEnv env-variable [env-variable] ...	svdh	B	
传送shell中的环境变量			
PidFile filename	logs/httpd.pid	s	M
服务器用于记录父进程(监控进程)PID的文件			
[ProtocolEcho On	Off](#calibre_link-833)	sv	X
Turn the echo server on or off			
<Proxy wildcard-url> ...</Proxy>	sv	E	
应用于所代理资源的容器			
[ProxyBadHeader IsError	Ignore	StartBody](#calibre_link-834)	IsE
确定如何处理不合法的应答头			
[ProxyBlock *	word	host	do
设置被代理屏蔽的语句、主机、域			
ProxyDomain Domain	sv	E	
代理请求的默认域名			
[ProxyErrorOverride On	Off](#calibre_link-835)	Off	sv
覆盖代理内容的错误页			
ProxyIOBufferSize bytes	8192	sv	E
内部缓冲区大小			
<ProxyMatch regex> ...</ProxyMatch>	sv	E	
应用于匹配正则表达式的代理资源的容器			
ProxyMaxForwards number	10	sv	E
转发请求的最大代理数目			

转发请求的最大代理数目			
[ProxyPass [path] !	url key=value key=value ...]]	svd	E
将一个远端服务器映射到本地服务器的URL空间中			
ProxyPassReverse [path] url	svd	E	
调整由反向代理服务器发送的HTTP应答头中的URL			
ProxyPassReverseCookieDomain internal-domain public-domain	svd	E	
Adjusts the Domain string in Set-Cookie headers from a reverse-proxied server			
ProxyPassReverseCookiePath internal-path public-path	svd	E	
Adjusts the Path string in Set-Cookie headers from a reverse-proxied server			
[ProxyPreserveHost On	Off](#calibre_link-670)	Off	sv
使用进入的HTTP请求头来发送代理请求			
ProxyReceiveBufferSize bytes	0	sv	E
代理HTTP和FTP连接的接收缓冲区大小(字节)			
ProxyRemote match remote-server	sv	E	
用于处理某些特定请求的远端代理			
ProxyRemoteMatch regex remote-server	sv	E	
处理匹配正则表达式的请求的远端代理			
[ProxyRequests On	Off](#calibre_link-407)	Off	sv
启用正向(标准)代理请求			
ProxyTimeout seconds	300	sv	E
代理请求的网络超时			
[ProxyVia On	Off	Full	Blc (#c 84
控制代理对Via应答头的使用			

ReadmeName filename	svdh	B	
Name of the file that will be inserted at the end of the index listing			
ReceiveBufferSize bytes	0	s	M
TCP接收缓冲区大小(字节)			
Redirect [status] URL-path URL	svdh	B	
发送一个外部重定向使客户端重定向到一个不同的URL			
RedirectMatch [status] regex URL	svdh	B	
基于正则表达式匹配对当前的URL发送一个外部重定向			
RedirectPermanent URL-path URL	svdh	B	
发送一个外部永久重定向使客户端重定向到一个不同的URL			
RedirectTemp URL-path URL	svdh	B	
发送一个外部临时重定向使客户端重定向到一个不同的URL			
RemoveCharset extension [extension] ...	vdh	B	
删除任何给定的扩展名与内容字符集之间的关联			
RemoveEncoding extension [extension] ...	vdh	B	
删除任何给定的扩展名与内容编码方式之间的关联			
RemoveHandler extension [extension] ...	vdh	B	
删除任何指定扩展名与处理器之间的关联			
RemoveInputFilter extension [extension] ...	vdh	B	
删除指定扩展名与输入过滤器之间的关联			
RemoveLanguage extension [extension] ...	vdh	B	
删除指定的扩展名与内容语言之间的关联			
RemoveOutputFilter extension			

RemoveOutputFilter extension [extension] ...	vdh	B	
删除指定扩展名与输出过滤器之间的关联			
RemoveType extension [extension] ...	vdh	B	
删除指定扩展名与内容类型之间的关联			
[RequestHeader set	append	add	uns [va
配置HTTP请求头			
Require entity-name [entity-name] ...	dh	C	
指定哪些认证用户允许访问该资源			
RewriteBase *URL-path*	dh	E	
Sets the base URL for per-directory rewrites			
RewriteCond *TestString CondPattern*	svdh	E	
Defines a condition under which rewriting will take place			
[RewriteEngine on	off](#calibre_link-629)	off	svc
Enables or disables runtime rewriting engine			
RewriteLock *file-path*	s	E	
Sets the name of the lock file used for `RewriteMap` synchronization			
RewriteLog *file-path*	sv	E	
Sets the name of the file used for logging rewrite engine processing			
RewriteLogLevel *Level*	0	sv	E
Sets the verbosity of the log file used by the rewrite engine			
RewriteMap *MapName MapType:MapSource*	sv	E	
Defines a mapping function for key-lookup			
RewriteOptions Options	svdh	E	

Sets some special options for the rewrite engine			
RewriteRule *Pattern Substitution*	svdh	E	
Defines rules for the rewriting engine			
[RLimitCPU seconds	max [seconds	max]](#calibre_link-479)	svc
限制Apache子进程派生的进程占用CPU的最大秒数			
[RLimitMEM bytes	max [bytes	max]](#calibre_link-480)	svc
限制由Apache子进程派生的进程占用的最大内存字节数			
[RLimitNPROC number	max [number	max]](#calibre_link-481)	svc
限制由Apache子进程派生的进程所派生的进程数目			
[Satisfy Any	All](#calibre_link-644)	All	dh
主机级别的访问控制和用户认证之间的相互关系			
ScoreBoardFile file-path	logs/apache_status	s	M
存储子进程协调数据(coordination data)的文件			
Script method cgi-script	svd	B	
对特定的请求方法激活一个CGI脚本			
[ScriptAlias URL-path file-path	directory-path](#calibre_link-219)	sv	B
映射一个URL到文件系统并视之为CGI脚本			
[ScriptAliasMatch regex file-path	directory-path](#calibre_link-518)	sv	B
使用正则表达式映射一个URL到文件系统并视之为CGI脚本			
[ScriptInterpreterSource Registry	Registry-Strict	Script](#calibre_link-371)	Sc

定位CGI脚本解释器			
ScriptLog file-path	sv	B	
CGI脚本错误日志文件的位置			
ScriptLogBuffer bytes	1024	sv	B
记入日志文件的PUT或POST请求头的最大数量			
ScriptLogLength bytes	10385760	sv	B
日志文件的大小限制(字节)			
ScriptSock file-path	logs/cgisock	sv	B
用来与CGI守护进程通信的套接字文件名前缀			
SecureListen [IP-address:]portnumber Certificate-Name [MUTUAL]	s	B	
在指定端口启用SSL加密			
SendBufferSize bytes	0	s	M
TCP发送缓冲区大小(字节)			
[ServerAdmin email-address	URL](#calibre_link-45)	sv	C
服务器返回给客户端的错误信息中包含的管理员邮件地址			
ServerAlias hostname [hostname] ...	v	C	
匹配一个基于域名的虚拟主机的别名			
ServerLimit number	s	M	
服务器允许配置的进程数上限			
ServerName fully-qualified-domain-name[:port]	sv	C	
服务器用于辨识自己的主机名和端口号			
ServerPath URL-path	v	C	
为兼容性不好的浏览器访问基于域名的虚拟主机保留的URL路径名			
ServerRoot directory-path	/usr/local/apache	s	C
安装服务器的基础目录			
[ServerSignature On	Off	EMail](#calibre_link-	Off

配置服务器生成页面的页脚			
[ServerTokens Major	Minor	Min[imal]	Pr
配置"Server:"应答头			
SetEnv env-variable value	svdh	B	
设置环境变量			
SetEnvIf *attribute regex [!]env-variable*[=*value*] [[!]*env-variable*[=*value*]] ...	svdh	B	
根据客户端请求属性设置环境变量			
SetEnvIfNoCase *attribute regex [!]env-variable*[=*value*] [[!]*env-variable*[=*value*]] ...	svdh	B	
根据大小写无关的客户端请求属性设置环境变量			
[SetHandler handler-name	None](#calibre_link-97)	svdh	C
强制所有匹配的文件被一个指定的处理器处理			
SetInputFilter filter[;filter...]	svdh	C	
设置处理客户端请求和POST输入时使用的过滤器			
SetOutputFilter filter[;filter...]	svdh	C	
设置用于处理服务器输出应答的过滤器			
SSIEndTag tag	"-->"	sv	B
String that ends an include element			
SSIErrorMsg message	"[an error occurred +	svdh	B
Error message displayed when there is an SSI error			
SSIStartTag tag	"<!--#"	sv	B
String that starts an include element			
SSITimeFormat formatstring	"%A, %d-%b-%Y %H:%M +	svdh	B
Configures the format in which date strings are displayed			

SSIUndefinedEcho string	"(none)"	svdh	B
String displayed when an unset variable is echoed			
SSLCACertificateFile *file-path*	sv	E	
File of concatenated PEM-encoded CA Certificates for Client Auth			
SSLCACertificatePath *directory-path*	sv	E	
Directory of PEM-encoded CA Certificates for Client Auth			
SSLCADNRequestFile *file-path*	sv	E	
File of concatenated PEM-encoded CA Certificates for defining acceptable CA names			
SSLCADNRequestPath *directory-path*	sv	E	
Directory of PEM-encoded CA Certificates for defining acceptable CA names			
SSLCARevocationFile *file-path*	sv	E	
File of concatenated PEM-encoded CA CRLs for Client Auth			
SSLCARevocationPath *directory-path*	sv	E	
Directory of PEM-encoded CA CRLs for Client Auth			
SSLCertificateChainFile *file-path*	sv	E	
File of PEM-encoded Server CA Certificates			
SSLCertificateFile *file-path*	sv	E	
Server PEM-encoded X.509 Certificate file			
SSLCertificateKeyFile *file-path*	sv	E	
Server PEM-encoded Private Key file			
SSLCipherSuite *cipher-spec*	ALL:!ADH:RC4+RSA:+H +	svdh	E
Cipher Suite available for			

SSLCryptoDevice *engine*	builtin	s	E
Enable use of a cryptographic hardware accelerator			
[SSLEngine on	off	optional](#calibre_link-860)	off
SSL Engine Operation Switch			
SSLHonorCiperOrder *flag*	sv	E	
Option to prefer the server's cipher preference order			
SSLMutex *type*	none	s	E
Semaphore for internal mutual exclusion of operations			
[SSLOptions [+	-]*option* ...](#calibre_link-863)	svdh	E
Configure various SSL engine run-time options			
SSLPassPhraseDialog *type*	builtin	s	E
Type of pass phrase dialog for encrypted private keys			
[SSLProtocol [+	-]*protocol* ...](#calibre_link-865)	all	sv
Configure usable SSL protocol flavors			
SSLProxyCACertificateFile *file-path*	sv	E	
File of concatenated PEM-encoded CA Certificates for Remote Server Auth			
SSLProxyCACertificatePath *directory-path*	sv	E	
Directory of PEM-encoded CA Certificates for Remote Server Auth			
SSLProxyCARevocationFile *file-path*	sv	E	
File of concatenated PEM-encoded CA CRLs for Remote Server Auth			
SSLProxyCARevocationPath *directory-path*	sv	E	

Directory of PEM-encoded CA CRLs for Remote Server Auth			
SSLProxyCipherSuite *cipher-spec*	ALL:!ADH:RC4+RSA:+H +	svdh	E
Cipher Suite available for negotiation in SSL proxy handshake			
[SSLProxyEngine on	off](#calibre_link-871)	off	sv
SSL Proxy Engine Operation Switch			
SSLProxyMachineCertificateFile *filename*	s	E	
File of concatenated PEM-encoded client certificates and keys to be used by the proxy			
SSLProxyMachineCertificatePath *directory*	s	E	
Directory of PEM-encoded client certificates and keys to be used by the proxy			
[SSLProxyProtocol [+	-]*protocol* ...](#calibre_link-874)	all	sv
Configure usable SSL protocol flavors for proxy usage			
SSLProxyVerify *level*	none	svdh	E
Type of remote server Certificate verification			
SSLProxyVerifyDepth *number*	1	svdh	E
Maximum depth of CA Certificates in Remote Server Certificate verification			
SSLRandomSeed *context source [bytes]*	s	E	
Pseudo Random Number Generator (PRNG) seeding source			
SSLRequire *expression*	dh	E	
Allow access only when an arbitrarily complex boolean expression is true			
SSLRequireSSL	dh	E	
Deny access when SSL is not used			

for the HTTP request			
SSLSessionCache *type*	none	s	E
Type of the global/inter-process SSL Session Cache			
SSLSessionCacheTimeout *seconds*	300	sv	E
Number of seconds before an SSL session expires in the Session Cache			
SSLUserName *varname*	sdh	E	
Variable name to determine user name			
SSLVerifyClient *level*	none	svdh	E
Type of Client Certificate verification			
SSLVerifyDepth *number*	1	svdh	E
Maximum depth of CA Certificates in Client Certificate verification			
StartServers number	s	M	
服务器启动时建立的子进程数			
StartThreads number	s	M	
服务器启动时建立的线程数			
SuexecUserGroup *User Group*	sv	E	
运行CGI程序的用户和组			
ThreadLimit number	s	M	
每个子进程可配置的线程数上限			
ThreadsPerChild number	s	M	
每个子进程建立的线程数			
ThreadStackSize size	s	M	
处理客户端连接的线程使用的栈尺寸(字节)			
TimeOut seconds	300	s	C
服务器在断定请求失败前等待的秒数			
[TraceEnable [on	off	extended]] (#calibre_link-882)	on
确定如何处理TRACE请求			

[TransferLog file	pipe](#calibre_link-276)	sv	B
指定日志文件的位置			
TypesConfig file-path	conf/mime.types	s	B
指定mime.types文件的位置			
UnsetEnv env-variable [env-variable] ...	svdh	B	
删除一个环境变量			
[UseCanonicalName On	Off	DNS](#calibre_link-102)	Of
配置服务器如何确定它自己的域名和端口			
[UseCanonicalPhysicalPort On	Off](#calibre_link-469)	Off	sv
配置服务器如何确定自己的名字和端口			
User unix-userid	#-1	s	M
实际服务于请求的子进程运行时的用户			
UserDir *directory-filename*	sv	B	
用户网站目录的位置			
[VirtualDocumentRoot *interpolated-directory*	none](#calibre_link-676)	none	sv
对于给定的基于名称的虚拟主机动态配置根文档目录			
[VirtualDocumentRootIP *interpolated-directory*	none](#calibre_link-675)	none	sv
对于给定的基于IP的虚拟主机动态配置根文档目录			
<VirtualHost addr[:port] [addr[:port]] ...> ... </VirtualHost>	s	C	
包含仅作用于指定主机名或IP地址的指令			
[VirtualScriptAlias *interpolated-directory*	none](#calibre_link-677)	none	sv
对于给定的基于名称的虚拟主机动态配置CGI目录			
[VirtualScriptAliasIP *interpolated-*	none](#calibre_link-883)	none	sv

对于给定的基于IP的虚拟主机动态配置CGI目录			
Win32DisableAcceptEx	s	M	
使用accept()代替AcceptEx()接受网络链接			
[XBitHack on	off	full] (#calibre_link-285)	off
Parse SSI directives in files with the execute bit set			

致谢

本手册是在原来尚未完成的“Apache 2.0手册中文版翻译项目”的基础上，根据新的 Apache 2.2 文档进行修订、更新、补充的结果，没有他们之前辛勤劳动积累的成果，单靠一人之力是很难完成的[目前尚有部分未翻译]。所以首先应当感谢哪个项目的各位自愿者：kajaa 、biAji 、fei 、suncjs 、Daniel 、flytosea 、forehead 。

感谢LinuxFans.Org上热心的 sejishikong 兄提供网络空间[网通]。

感谢LinuxSir.Org上热心的 bingzhou 兄提供网络空间[电信]。

感谢热心的 乔章池 兄制作 chm 和 pdf 版本。

译者声明

- 由于译者水平有限，因此不能保证译文准确无误，请在使用中自行鉴别。
- 本手册虽然不是软件，但是本着GPL的精神发布。
- 中译版每个页面的版权分别属于每个页面的译者。
- 任何人都可以自由使用、分发、转载，包括对本文档进行各种商业性的或是非商业性的复制和再分发，但必须保留译者署名，亦不得对声明中的任何条款作任何形式的修改，也不得附加任何其它的条件。
- 您可以自由链接、下载、传播此文档，或者放置在您的网站上，甚至作为产品的一部分发行。但前提是必须保证全文完整转载，包括完整的版权信息和作译者声明。这里“完整”的含义是，不能进行任何删除/增添/注解。若有删除/增添/注解，必须逐段明确声明那些部分并非本文档的一部分。

最新版本

译者将会不定期的根据用户提交的修改意见以及官方手册进行更新，最新的版本可以在如下位置获得：

- 电信用户请访问：Apache 2.2 中文手册 http://lamp.linux.gov.cn/Apache/ApacheMenu/index.html
- 网通用户请访问：Apache 2.2 中文手册 http://www.dogdoghome.com/lamp/Apache/ApacheMenu/index.html

打包下载：

- 电信用户：rar bz2 zip pdf chm
- 网通用户：rar bz2 zip pdf chm

招募自愿者

一方面由于译者水平有限，不可能精通Apache的所有方面，为了保证翻译质量，某些不熟悉的章节不敢冒然翻译；另一方面由于译者人数有限、精力也有限，单靠一人或寥寥数人之力很难完成所有内容。但是“众人拾柴火焰高”，我们期待翻译作风严谨、热爱开源的兄弟姐妹加入到这个项目中来，为促进Apache的应用和开源运动在中国的发展尽上一份绵薄之力。同时也了却原2.0项目组的夙愿。

除此以外，如果你发现了已经完成的译文中有错误，也请您指出，哪怕是错别字也好，任何提高译文质量的建议我们都将虚心接纳。

如果你希望加入这个队伍，或者发现译文中有错误，请与金步国联系，QQ：70171448 ；MSN：csfrank122@hotmail.com

目前的译者队伍如下：

- 金步国

译者的其他作品

本手册的译者都是热爱开源的兄弟姐妹，都十分愿意与他人共享劳动成果，如果你对他们的其他翻译作品或者技术文章有兴趣，可以在如下两个位置查看现有作品的列表(随时更新)：

- 电信用户请访问：作品列表
- 网通用户请访问：作品列表

模块索引

以下是Apache发行版中所有模块的列表。字母顺序列表参见指令索引和稍微详细一点的指令速查。 另外还有描述指令的术语和描述模块的术语以供参考。

核心功能和多路处理模块

core

Apache HTTP服务器核心提供的功能，始终有效。

mpm_common

收集了被多个多路处理模块(MPM)实现的公共指令。

beos

专门针对BeOS优化过的多路处理模块(MPM)

event

一个标准 `worker` MPM的实验性变种。

mpm_netware

Novell NetWare优化过的线程化的多路处理模块(MPM)

mpmt_os2

专门针对OS/2优化过的混合多进程多线程多路处理模块(MPM)

prefork

一个非线程型的、预派生的MPM

mpm_winnt

用于Windows NT/2000/XP/2003 系列的MPM

worker

线程型的MPM，实现了一个混合的多线程多处理MPM，允许一个子进程中包含多个线程。

其它普通模块

mod_actions

基于媒体类型或请求方法，为执行CGI脚本而提供

mod_alias

提供从文件系统的不同部分到文档树的映射和URL重定向

mod_asis

发送自己包含HTTP头内容的文件

mod_auth_basic

使用基本认证

mod_auth_digest

使用MD5摘要认证(更安全，但是只有最新的浏览器才支持)

mod_authn_alias

基于实际认证支持者创建扩展的认证支持者，并为它起一个别名以便于引用

mod_authn_anon

提供匿名用户认证支持

mod_authn_dbd

使用SQL数据库为认证提供支持

mod_authn_dbm

使用DBM数据库为认证提供支持

mod_authn_default

在未正确配置认证模块的情况下简单拒绝一切认证信息

mod_authn_file

使用纯文本文件为认证提供支持

mod_authnz_ldap

允许使用一个LDAP目录存储用户名和密码数据库来执行基本认证和授权

mod_authz_dbm

使用DBM数据库文件为组提供授权支持

mod_authz_default

在未正确配置授权支持模块的情况下简单拒绝一切授权请求

mod_authz_groupfile

使用纯文本文件为组提供授权支持

mod_authz_host

供基于主机名、IP地址、请求特征的访问控制

mod_authz_owner

基于文件的所有者进行授权

mod_authz_user

基于每个用户提供授权支持

mod_autoindex

自动对目录中的内容生成列表，类似于"ls"或"dir"命令

mod_cache

基于URI键的内容动态缓冲(内存或磁盘)

mod_cern_meta

允许Apache使用CERN httpd元文件，从而可以在发送文件时对头进行修改

mod_cgi

在非线程型MPM(`prefork`)上提供对CGI脚本执行的支持

mod_cgid

在线程型MPM(`worker`)上用一个外部CGI守护进程执行CGI脚本

mod_charset_lite

允许对页面进行字符集转换

mod_dav

允许Apache提供DAV协议支持

mod_dav_fs

为 `mod_dav` 访问服务器上的文件系统提供支持

mod_dav_lock

为 `mod_dav` 锁定服务器上的文件提供支持

mod_dbd

管理SQL数据库连接，为需要数据库功能的模块提供支持

mod_deflate

压缩发送给客户端的内容

mod_dir

指定目录索引文件以及为目录提供"尾斜杠"重定向

mod_disk_cache

基于磁盘的缓冲管理器

mod_dumpio

将所有I/O操作转储到错误日志中

mod_echo

一个很简单的协议演示模块

mod_env

允许Apache修改或清除传送到CGI脚本和SSI页面的环境变量

mod_example

一个很简单的Apache模块API演示模块

mod_expires

允许通过配置文件控制HTTP的"`Expires:`"和"`Cache-Control:`"头内容

mod_ext_filter

使用外部程序作为过滤器

mod_file_cache

提供文件描述符缓存支持，从而提高Apache性能

mod_filter

根据上下文实际情况对输出过滤器进行动态配置

mod_headers

允许通过配置文件控制任意的HTTP请求和应答头信息

mod_ident

实现RFC1413规定的ident查找

mod_imagemap

处理服务器端图像映射

mod_include

实现服务端包含文档(SSI)处理

mod_info

生成Apache配置情况的Web页面

mod_isapi

仅限于在Windows平台上实现ISAPI扩展

mod_ldap

为其它LDAP模块提供LDAP连接池和结果缓冲服务

mod_log_config

允许记录日志和定制日志文件格式

mod_log_forensic

实现"对比日志"，即在请求被处理之前和处理完成之后进行两次记录

mod_logio

对每个请求的输入/输出字节数以及HTTP头进行日志记录

mod_mem_cache

基于内存的缓冲管理器

mod_mime

根据文件扩展名决定应答的行为(处理器/过滤器)和内容(MIME类型/语言/字符集/编码)

mod_mime_magic

通过读取部分文件内容自动猜测文件的MIME类型

mod_negotiation

提供内容协商支持

mod_nw_ssl

仅限于在NetWare平台上实现SSL加密支持

mod_proxy

提供HTTP/1.1的代理/网关功能支持

mod_proxy_ajp

`mod_proxy` 的扩展，提供Apache JServ Protocol支持

mod_proxy_balancer

`mod_proxy` 的扩展，提供负载平衡支持

mod_proxy_connect

`mod_proxy` 的扩展，提供对处理HTTP `CONNECT` 方法的支持

mod_proxy_ftp

`mod_proxy` 的FTP支持模块

mod_proxy_http

`mod_proxy` 的HTTP支持模块

mod_rewrite

一个基于一定规则的实时重写URL请求的引擎

mod_setenvif

根据客户端请求头字段设置环境变量

mod_so

允许运行时加载DSO模块

mod_speling

自动纠正URL中的拼写错误

mod_ssl

使用安全套接字层(SSL)和传输层安全(TLS)协议实现高强度加密传输

mod_status

生成描述服务器状态的Web页面

mod_suexec

使用与调用web服务器的用户不同的用户身份来运行CGI和SSI程序

mod_unique_id

为每个请求生成唯一的标识以便跟踪

mod_userdir

允许用户从自己的主目录中提供页面(使用"/~username")

mod_usertrack

使用Session跟踪用户(会发送很多Cookie)，以记录用户的点击流

mod_version

提供基于版本的配置段支持

mod_vhost_alias

提供大批量虚拟主机的动态配置支持

站点导航

本页列出了当前所有可用的Apache HTTP Server Version 2.2 文档。

发行说明

- 致谢与译者声明
- 从 1.3 升级到 2.0
- 从 2.0 升级到 2.2
- Apache 2.1/2.2 的新特性
- Apache 2.0 的新特性
- Apache License

Apache HTTP服务器的使用

- 编译与安装 Apache
- 启动Apache
- 停止和重新启动服务器
- 配置文件
- Directory、Location、Files 配置段的工作方式
- 缓冲指南
- 服务器全局配置
- 日志文件
- 从URL到文件系统的映射
- 安全方面的提示
- 动态共享对象(DSO)支持
- 内容协商
- 自定义错误响应
- 设置Apache绑定的地址和端口
- 多路处理模块(MPM)
- Apache的环境变量
- Apache处理器的使用
- 过滤器
- suEXEC支持
- 性能方面的提示
- URL重写指南

Apache虚拟主机文档

- 概述
- 基于主机名的虚拟主机
- 基于IP地址的虚拟主机
- 动态配置大量虚拟主机
- 虚拟主机的例子
- 虚拟主机匹配的深入讨论
- 文件描述符的限制
- 关于DNS和Apache

Apache服务器常见问题解答

- 概述(背景知识/支持/错误信息)

Apache的SSL/TLS加密

- 概述
- SSL/TLS 加密：入门
- SSL/TLS 加密：兼容性
- SSL/TLS 加密：如何...
- SSL/TLS 加密：常见问题解答

指导性文档

- 概述
- 认证、授权、访问控制
- CGI动态页面
- 服务器端包含(SSI)入门
- .htaccess文件
- 用户网站目录

针对特定平台的说明

- 概述
- 在Microsoft Windows中使用Apache
- 在Microsoft Windows中编译Apache
- 在Novell NetWare中运行Apache

Apache HTTP服务器和支持程序

杂项文档

Apache模块

- mod_filter
- mod_headers
- mod_ident
- mod_imagemap
- mod_include
- mod_info
- mod_isapi
- mod_ldap
- mod_log_config
- mod_log_forensic
- mod_logio
- mod_mem_cache
- mod_mime
- mod_mime_magic
- mod_negotiation
- mod_nw_ssl
- mod_proxy
- mod_proxy_ajp
- mod_proxy_balancer
- mod_proxy_connect
- mod_proxy_ftp
- mod_proxy_http
- mod_rewrite
- mod_setenvif
- mod_so
- mod_speling
- mod_ssl
- mod_status
- mod_suexec
- mod_unique_id
- mod_userdir
- mod_usertrack
- mod_version
- mod_vhost_alias

开发者文档

- 概述
- Apache API 备忘录
- APR中内存分配的调试

- Apache2.0文档制作
- Apache2.0中的Hook函数
- 将模块从Apache1.3转化为Apache2.0
- Apache2.0中对请求的处理
- Apache2.0中处理器的运作

词汇和索引

- 词汇表
- 模块索引
- 指令索引
- 指令速查